(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 August 2003 (14.08.2003)

PCT

(10) International Publication Number
WO 03/067810 A1

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US03/03652

(22) International Filing Date: 7 February 2003 (07.02.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/072,683 8 February 2002 (08.02.2002) US

(71) Applicant *(for all designated States except US)*: **NETSCREEN TECHNOLOGIES, INC.** [US/US]; 350 Oakmead Parkway, Sunnyvale, CA 94085 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **GURUSWAMY, Kowsik** [IN/US]; 529 E. McKinley Avenue, Sunnyvale, CA 94086 (US). **ZUK, Nir** [IL/US]; 564 S. Oak Park Way, Redwood City, CA 94062 (US).

(74) Agents: **KIRKLAND, Mark, D.** et al.; Fish & Richardson P.C., 500 Arguello Street, Suite 500, Redwood City, CA 94063 (US).

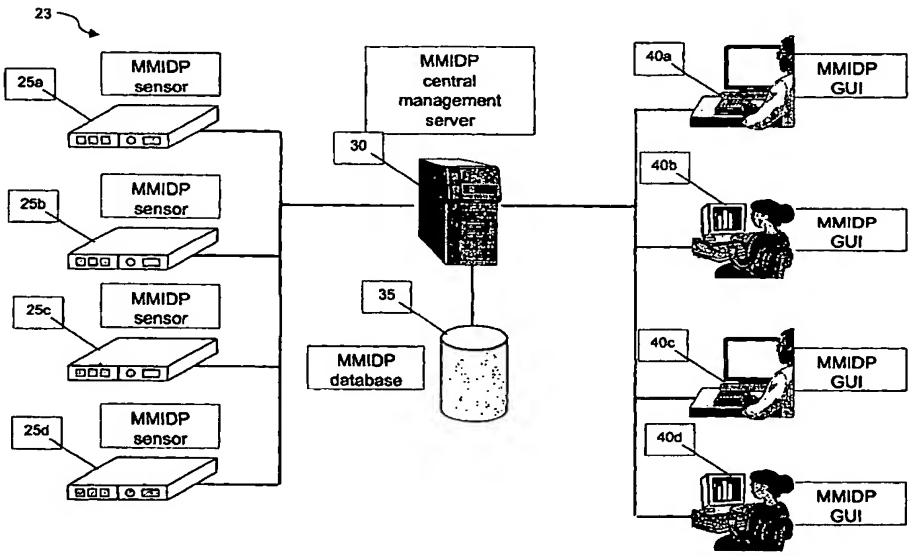
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: MULTI-METHOD GATEWAY-BASED NETWORK SECURITY SYSTEMS AND METHODS



(57) Abstract: Systems and methods for detecting and preventing network security breaches (23) are described. The systems and methods present a gateway-based packet-forwarding network security solution to not only detect security breaches but also prevent from by directly dropping suspicious packets and connections. The systems and methods employ multiple techniques to detect and prevent network security breaches (30), including stateful signature detection, traffic signature detection, and protocol anomaly detection (35).

WO 03/067810 A1

Multi-Method Gateway-Based Network Security Systems and Methods

Field of the Invention

This invention relates generally to network security systems and methods for detecting and preventing security breaches on a network. More specifically, the present invention provides gateway-based packet-forwarding network security systems and methods to not only detect security breaches on the network but also prevent them by directly dropping suspicious packets and connections. These systems and methods employ multiple techniques to detect and prevent intrusions, including stateful signature detection, traffic signature detection, and protocol anomaly detection.

Background

The explosion of the Internet has revolutionized the ways in which information is disseminated and shared. At any given time, massive amounts of information are exchanged electronically by millions of individuals worldwide using the Internet but also for engaging in a wide variety of activities, including communication, commercial transactions, and entertainment.

The Internet breaks down traditional geographical barriers by not requiring a dedicated end-to-end connection for communicating information between a source and a destination network host. Instead, Internet traffic is split up into units of information called "packets" that are routed dynamically through the network based on the most efficient route between the source and the destination at any given moment. Each of these packets includes a "header", which indicates the source from which the information originates and the destination to which it is being sent as well as other information necessary for routing the packets through the network. The source and destination are identified by means of an "IP address", a 32-bit number associated to each network host.

Packet headers conform to a set of shared "protocols" used in all Internet transmissions. Those protocols are the set of conventions that determine how information will be exchanged, often between computers from different manufacturers and running different operating systems. Internet protocols specify how the network moves data, handles errors, and allows information to be sent, received, and understood. The most fundamental protocol is called "Internet

protocol", or IP, responsible for the formatting and delivery of packets across the network. Transport protocols such as UDP, TCP, and RTP, are used on top of IP to ensure that the data in the packets is received correctly, with the TCP protocol further guaranteeing that the packets are received reliably. Additional features and capabilities are provided by special-purpose protocols that are used together with the IP and transport protocols.

While the Internet protocol structure provides unparalleled benefits to users, it also facilitates unlawful activity by providing a vast, inexpensive, and potentially anonymous way for breaching security on any Internet host, including private networks of which those hosts are a part. Despite the number of potential network security vulnerabilities, current network security technologies are inadequate and ineffective to detect and prevent the increasingly sophisticated and numerous network security breaches. Examples of existing network security technologies range from operating system controls, password protection tools, and anti-virus software to more sophisticated technologies such as virtual private networks, firewalls, and intrusion detection systems.

Virtual private networks ("VPNs") are private networks established over any shared network, such as the Internet. VPNs attempt to maintain privacy through the use of security procedures involving authentication and encryption between any two VPN termination points, such as a router in a remote office, a laptop, a server application, and so on. In addition, VPNs often make use of secure tunneling protocols such as the developing standard Internet Protocol Security ("IPSec") that consists of a set of Internet security services for the IP layer, including authentication, packet integrity and confidentiality, and encryption key management. VPNs are typically integrated into firewall software to improve network security.

A firewall is a set of software programs located at a private network gateway that attempts to filter information flowing between the private network and a shared network such as the Internet. A firewall attempts to protect the private network resources from outsiders and to control the private network users' access to outside resources. There are four main types of firewalls in use today: packet filters, circuit-level gateways, application gateways, and stateful inspection. There also may be hybrid firewalls that are a combination of any two or more of all four firewall types.

Packet filtering firewalls compare header information in the incoming and outgoing IP packets on a private network against a table of rules or filters set up by

the network administrator to verify whether the packets meet the requirements in the table. If a packet does not conform to those rules, the firewall will reject the packet and the packet will not be forwarded to its destination. Header information examined by packet filtering firewalls typically includes source and destination addresses, protocol type, the network interface through which the packet enters, the direction of traffic, routing, and connection state, among others. For example, a packet filtering firewall may specify that any UDP packet coming from IP addresses ranging from 232.181.20.10 to 232.181.20.255 will not be allowed into the private network.

The security of a private network having a packet filtering firewall may be increased by using Network Address Translation ("NAT") within the firewall. NAT functions like a private branch exchange in a telephone system. All the source addresses of outgoing IP packets are rewritten to the IP address assigned to the firewall to give the impression that the packets originated from the firewall rather than from the internal hosts of the private network protected by the firewall. Reply packets coming back are translated and forwarded to the appropriate host. With NAT, internal hosts are allowed to connect to hosts outside of the firewall but outside hosts cannot connect directly to the internal hosts since they are only aware of the IP address of the firewall.

Packet filtering firewalls are relatively inexpensive and do not interfere with network performance, but alone they cannot typically provide adequate security. Packet filtering rules become unmanageable in complex environments, provide no user authentication mechanisms, and are vulnerable to attacks such as IP spoofing. For example, if a hacker can figure out a trusted IP address, the hacker may forge an IP header to a harmful packet. Being unable to differentiate between a valid packet and a forged one, a packet filtering firewall would not reject the harmful packet.

Examples of packet filtering firewalls include the freely distributed software package IPFilter for UNIX-based operating systems, the freely distributed SINUS TCP/IP packet filter provided for the Linux operating system under a GNU general public license, and the protocol-based Personal Firewall PRO™ sold by Sygate Technologies, Inc., of Fremont, CA.

Another type of firewall referred to as a circuit-level firewall operates at the session layer of the network to validate TCP/IP sessions before opening a connection. Circuit-level firewalls allow TCP packets to pass through only after a packet handshake has taken place. A packet handshake starts with the source sending a

synchronize ("SYN") packet to the destination and ends with the destination sending a SYN packet and an acknowledgment ("ACK") packet back to the source. Circuit-level firewalls maintain a table of valid connections, which includes session state and sequence number information of the SYN and ACK packets, and allow packets to pass through when the network packet information matches an entry in the table. All packets transmitted after the handshake are allowed until the session is ended.

A circuit-level firewall maintains two connections per session, one between the source and the firewall and another between the firewall and the destination. As a result, all outgoing packets appear to have originated from the firewall similar to packet filtering firewalls with NAT, that is, direct contact between the source and the destination is prevented.

Circuit-level firewalls have good performance once the initial connections are established and offer a high degree of flexibility. However, they cannot examine the application-level content of the packets it is transmitting in any given connection. Once a connection has been established, any malicious application or packet can run across the connection.

Most circuit-level firewalls are implemented using the publicly available "SOCKS" networking protocol that enables hosts on one side of a SOCKS server to access hosts on the other side of the SOCKS server without requiring direct IP reachability. When an application client starts a session with an application server via a SOCKS server, the client first sends the SOCKS server a list of authentication methods it supports. The SOCKS firewall then compares these methods against the security policy defined by the network administrator, chooses an authentication method, sends a message to the client telling which authentication method to use, and finally, authenticates the client. After the client is authenticated, the SOCKS server establishes a virtual circuit between the client and the server to transmit all packets through the virtual circuit until the circuit is kept open. An example of a circuit-level firewall using SOCKS include Hummingbird SOCKS, provided by Hummingbird, Ltd., of Toronto, Canada.

To address the inherent security risk of circuit-level firewalls, application-level firewalls that operate at the application layer of the network were developed. Such firewalls run an application proxy server as an intermediary between the private network and the shared network for each allowed application, such as an FTP proxy, a HTTP proxy, a SMTP proxy for e-mail, and so on.

Application proxies are generally considered to be more secure than packet filtering or circuit-level firewalls. Similar to circuit-level firewalls, application proxies do not allow direct connections and force all packets to be screened for suitability. However, application proxies are typically slower than packet filtering or circuit-level firewalls because all packets have to be evaluated at the application layer, that is, every packet passing through an application proxy must undergo de-encapsulation/re-encapsulation before reaching its final destination. In addition, proxy servers may not have packet forwarding capabilities. Every new service requires a new proxy server, and because proxies are highly dependent on many other system components to operate correctly, such as operating systems, TCP/IP stacks, and runtime libraries, they are vulnerable to application-level security flaws and bugs.

Application-proxies are typically implemented with built-in packet filtering or stateful inspection capabilities. Examples include the VelociRaptor firewall sold by Symantec Corporation of Cupertino, CA, the Gauntlet firewall sold by Network Associates, Inc., of Santa Clara, CA, and the Sidewinder™ firewall sold by Secure Computing Corp., of San Jose, CA.

The performance of packet filtering firewalls, circuit-level firewalls, and application-proxies may be improved with the use of stateful inspection. Stateful inspection firewalls are essentially packet filtering firewalls that examine not just the packet header, but also information about the packet in all communication layers of the network, such as TCP packet headers, to analyze the network traffic that traverses it.

Such firewalls monitor the state of any given network connection and compile information about the connection in a state table. Each packet request coming out of the firewall is recorded in the state table so that incoming response packets are verified against the corresponding request packets in the state table. The decision on whether to reject a packet is therefore based not only on the packet filtering rules table but also on the context that has been established by prior packets that have passed through the firewall. A packet that is a genuine response to a request packet is passed on and all others are rejected. If a response packet does not arrive in a specified period of time, the connection is timed out.

A packet filtering firewall with stateful inspection also has the ability to examine a packet in order to allow certain types of commands within an application while disallowing others. For example, a stateful inspection firewall can allow the

FTP "get" command while disallowing the "put" command. In addition, stateful inspection firewalls incorporate dynamic filtering techniques to minimize the number of exposed network ports. With dynamic filtering, network ports are kept open only as required for packet flow based on packet header information, thereby reducing the attacks to open ports that are idle.

Examples of stateful inspection firewalls include the firewall described in U.S. Patent No. 5,606,668 and the firewall product called FireWall-1, sold by Check Point Software Technologies, Inc., of Redwood City, CA. FireWall-1 enables network administrators to define and implement a single, centrally managed security policy. The security policy is defined at a central management server by means of graphical user interface clients and downloaded to multiple enforcement points throughout the network. The security policy is defined in terms of security rules and network objects such as gateways, routers, and hosts. Packet header data is examined at all seven network layers and state information is kept of packets at all communication stages to verify IP addresses, port numbers, and any other information required to determine whether packets are permitted by the security policy.

State information is stored at a connections or state table that organizes packets according to their corresponding network connections, which are represented in the table by the source IP address, the source port, the destination IP address, the destination port, the IP protocol type, and other parameters including Kbuf, Type, Flags, and Timeout. When a packet is received by the firewall, the packet is checked against the connections table to see if there is an existing connection to which this packet belongs. If there is a connection, then the packet is forwarded to its network destination. If there is no matching connection in the state table for that specific packet, then the firewall compares it against the security policy to see if there is a match that allows the packet to pass. If there is, then the connection is added to the connections table and all subsequent packets belonging to that conversation will be forwarded along immediately, without being checked against the policy. As a result, a connection may be initially established with benign packets and then used to transmit malicious packets that will be accepted by the firewall. Another example of a stateful inspection firewall product is the PIX firewall sold by Cisco Systems, Inc., of San Jose, CA.

The sole role of the currently available firewalls is to enforce an organization's network access policies. Such access policies specify which hosts and protocols

represent good traffic, i.e., traffic that may be allowed in the network, and which ones do not. In other words, a firewall simply distinguishes good from bad traffic based on a pre-determined and static configuration embodied in the access policy. Firewalls are not capable of detecting and stopping network attacks. For example, once a firewall allows a HTTP connection, it will not be able to detect an attack against a web server carried over that connection. Furthermore, a firewall is not able to detect or prevent attacks made or appeared to be made from inside the firewall, such as the presence of a Trojan program inside the network that may be leaking confidential information to the outside.

To attempt to fill the gaps in network security left open by firewall products, "intrusion detection systems" have been developed and used in tandem with firewalls. An intrusion detection system ("IDS") collects information from a variety of system and network resources to analyze the information for signs of intrusion, i.e., attacks coming from outside the network, and misuse, i.e., attacks originating from inside the network. Intrusion detection systems can be placed inside or outside the firewall, with most network administrators choosing to place the IDS inside of the firewall as an extra layer of protection against misuse and intrusions undetected by the firewall.

There are three types of intrusion detection systems: desktop-based IDSs, host-based IDSs, and network-based IDSs. Desktop-based IDSs offer file-level protection by examining activity on individual systems, looking for potential attacks on files or registry entries. A desktop-based IDS may be useful for an individual user who connects to the Internet directly and is not part of any extensive network. A popular desktop-based IDS is the BlackICE Defender, sold by Internet Security Systems, Inc., of Atlanta, GA.

Host-based IDSs operate on a network host, such as a web or application server, tracking and analyzing entries in the host system's application and operating system logs to detect attacks and disallowed activity. Host-based IDSs are easy and inexpensive to deploy and do not require any additional hardware. Since they monitor events local to a host, they can detect attacks and disallowed activity that may not necessarily be seen by the network. However, because they consume considerable resources, they can adversely affect the host's performance. In addition, successful intrusions that gain high levels of privilege on the network may disable host-based IDSs and remove traces of their operation entirely. Examples of host-based IDSs

include the Intruder Alert IDS sold by Symantec Corporation of Cupertino, CA, and the Tripwire IDS sold by Tripwire, Inc., of Portland, OR.

Network-based IDSs ("NIDSs") are designed to protect multiple network hosts simultaneously by examining all the packets flowing through a network segment. NIDSs often consist of a set of single-purpose sensors or hosts placed at various points in a network. These units monitor network traffic, perform local analysis of that traffic and report attacks to a central management unit. Unlike firewalls, which typically only examine packet header information relating to IP addresses, ports, and protocol types, NIDSs may be designed to examine all the different flags and options that can exist in a network packet header as well as the packet data or payload, thereby detecting maliciously crafted packets that are designed to be overlooked by the firewall.

The most common network intrusion detection systems are signature-based systems and protocol anomaly also known as protocol analysis systems. Signature-based systems look for known attack patterns or signatures in network traffic. Signatures can be as simple as a character string that matches a portion of a network packet or as complex as a state machine. In general, a signature can be concerned with a process, such as the execution of a particular command, or an outcome, such as the acquisition of a root shell. When a signature-based NIDS finds a matching signature in a packet, it can then respond by taking a user-defined action, sending an alert, or performing additional logging of information.

Most signature-based NIDSs on the market use packet-signature detection, which means that they examine the raw bytes of every packet in a traffic flow to find a match for an attack pattern. As such, these systems have several drawbacks. First, since the entire traffic flow needs to be searched, network performance may be significantly diminished. Second, because more data are being searched, it is more likely for a signature to match irrelevant data and result in a false alarm. Third, since packet-signature NIDSs can only find attacks in a packet for which a signature is written, new and often very complicated attacks cannot be detected. And lastly, packet-signature NIDSs may fail to examine packets when the network traffic is too high.

Examples of signature-based NIDSs include the system described in U.S. Patent No. 6,279,113, the SecureIDS system, sold by Cisco Systems, Inc., of San

Jose, CA, the RealSecure system, sold by Internet Security Systems, Inc., of Atlanta, GA, and the NetProwler system, sold by Symantec Corporation, of Cupertino, CA.

In contrast to signature-based NIDSs that examine network traffic for some previously defined intrusions, "protocol anomaly" detection NIDSs examine network traffic for abnormalities in generally accepted Internet rules of communication. These rules are defined by open protocols, published standards, and vendor-defined specifications for communications between network devices. Once an irregularity is identified, it can be used to make network security decisions.

Protocol anomaly detection NIDSs provide several advantages over signature-based NIDSs, such as the ability to detect unknown attacks, including attacks that cannot be detected by signature matching, as well as known attacks that have been slightly modified to avoid detection from signature-based NIDSs. For example, protocol anomaly detection NIDSs can detect "FTP bounce" attacks that occur when an attacker tells the FTP server to open a connection to an IP address that is different from the user's address and "overflow" attacks that exploit the common buffer overflow programming error.

Nevertheless, there are attacks that conform to the protocol specifications and therefore cannot be detected by protocol anomaly detection systems. Such attacks require signatures or other methods of detection.

Examples of protocol anomaly detection NIDSs include BlackICE Guard, sold by Internet Security Systems, of Atlanta, GA, and ManHunt, sold by Recourse Technologies, Inc., of Redwood City, CA. An alternative to detecting abnormal network behavior as a result of protocol irregularities is suggested by StealthWatch, sold by Lancope, Inc., of Atlanta, GA. StealthWatch proposes a "flow-based" architecture to characterize the flow of packets between two hosts that are associated with a single service, such as using a web browser to access a single web server, or using an e-mail program to access a mail server.

While the NIDSs discussed above may improve a network's security, they have several drawbacks. First, false alarms are often produced by signature-based NIDSs that do not evaluate a signature within the context of the network traffic. For example, a signature-based NIDS may scan all e-mail messages for the string "I love you" to detect the infamous Internet worm that carries that name, which will create a false alarm with some personal e-mail. Second, most of the NIDSs discussed above use a single method of detection that is insufficient to comprehensively detect

intrusions. As such, false negatives are produced when the NIDSs do not detect an attack while it is occurring. For example, a protocol anomaly NIDS may generate a false negative when a hacker fools the NIDS to see network traffic differently from the target host so that the traffic can pass through the NIDS but ultimately infect the target host by using sophisticated packet and protocol tampering methods that cannot be detected by a protocol anomaly NIDS.

In addition, some NIDSs are not able to detect "port scans" and "network sweeps" used by attackers to identify potential security and system flaws that may be exploited. Port scans and network sweeps usually happen when an attacker attempts to determine which services are allowed on the network and to identify which network port would be a good entrance to an attack. The attacker may either try each and every port on a single network (port scan) or a certain port on an entire network (network sweep). That is, port scans and network sweeps are not attacks, but rather, indicators of imminent attacks. Neither signature-based nor anomaly detection NIDSs are able to identify port scans and network sweeps since a scan conforms to the particular network protocol being used to transmit the packet and the scan pattern does not appear within a particular network session.

A further drawback of most of the NIDSs discussed above is that they need to be individually managed and all sensor information resides on the sensor itself. That is, network security administrators need to access each individual sensor to activate or detect signatures, perform system management backups, and so on. As the number of sensors increases, management of the sensors becomes increasingly difficult, especially considering the often incomplete logs that are generated. In the event of failure of any sensor, the replacement sensor has to be reconfigured and tuned to match the original sensor.

Additionally, NIDSs cannot directly prevent attacks. NIDSs work as passive intrusion detection mechanisms, but are not capable to prevent attacks from occurring. When an attack is occurring on a network, these systems can notify a network security administrator to take action after the attack has already taken place but cannot prevent the attack itself. NIDSs do not sit directly in the path of traffic and cannot actively react to suspend a network connection being attacked or even redirect the intruding packets to a safer or more secure system.

An attempt to address this problem is described in U.S. Patent No. 6,119,236, which proposes to have an NIDS direct a firewall to take action if an attack is detected

to prevent the attack from spreading. That is, the NIDS does not directly prevent the attack, but simply interrupts it so that the attack may not become any worse. In doing so, the NIDS may inadvertently interrupt valid network traffic. For example, if an offending hacker is using a major Internet service provider IP address to attack the network and the NIDS system notifies the firewall to block the packets coming from this IP address, all users of the Internet service provider, malicious or not, will be denied network access.

Another proposal to address some of the deficiencies of current NIDSs is to make use of TCP reset packets to prevent TCP attacks. When a NIDS device detects a TCP attack, it sends a TCP reset packet to both the source and the destination network hosts to reset the TCP connection and prevent the attack from occurring. That is, this NIDS also does not directly prevent the attack, but simply interrupts it so that the attack may not become any worse. However, there are several problems with this approach. First, it takes a period of time for the NIDS to determine that an intrusion has been attempted and that a reset packet should be sent. During this period, the intruding packet and most likely some of the packets that follow it, may be transferred to the target network and reached the destination host. As a result, any TCP reset packet that is sent upon detection may be too late. Second, TCP reset packets are only available for the TCP protocol and cannot therefore be used to prevent attacks taking place using UDP or other connectionless protocols. And lastly, since a TCP reset packet must carry a valid sequence number within a small receiver window, a sophisticated attacker can transmit its intruding packets to have the server's receiver window change so rapidly that the NIDS will have difficulty in determining which sequence number to put in the TCP reset packet and fail to prevent the attack.

No firewall or NIDSs product, either alone or working in tandem, is able to examine packets allowed onto a network and react to disallowed packets or activity by directly dropping those packets or closing the connection. In addition, there is no hybrid NIDS that integrates signature detection, protocol anomaly detection, and other sophisticated methods such as traffic signature detection to achieve higher intrusion detection accuracy and thus reduce the rate of false positives and false negatives. There also is no NIDS that provides a centralized, policy-based management solution to control all the NIDS sensors. As a result, attempting to secure a network using technology and products available today can be impractical, if not impossible.

Summary of the Invention

In view of the foregoing, it is an object of the present invention to provide network security systems and methods capable of accurately and comprehensively detecting and preventing network security breaches with low false alarm rates.

It is also an object of the present invention to provide network systems and methods that can examine packets allowed onto a network and react to disallowed packets or activity by directly dropping those packets or closing the connection.

It is also an object of the present invention to provide network security systems and methods that integrate stateful signature detection, traffic signature detection, protocol anomaly detection as well as other methods to detect and prevent network security breaches.

It is a further object of the present invention to provide network security systems and methods that enable a network security administrator to centrally manage all the network intrusion detection sensors placed on the network.

These and other objects of the present invention are accomplished by providing multi-method network security systems and methods to detect and prevent network security breaches with low false alarm rates based on stateful signature detection, traffic signature detection, and protocol anomaly detection. The multi-method network security systems, hereinafter referred to as the "MMIDP system", consists of a software and hardware solution placed directly in the path of network traffic to drop any incoming or outgoing suspicious packets before they reach network hosts or the outside network. The MMIDP system may be used by itself or in conjunction with a firewall.

The systems and methods of the present invention have been advantageously incorporated into a preferred example of an MMIDP with four main components: (1) a network intrusion detection and prevention sensor; (2) a network intrusion detection and prevention central management server; (3) a network intrusion detection and prevention central database; and (4) a network intrusion detection and prevention graphical user interface.

The network intrusion detection and prevention sensor consists of a hardware appliance that may be placed at multiple gateway points in the path of network traffic. A given sensor may operate in gateway mode to drop any incoming or outgoing suspicious packet before it reaches the network hosts or the outside network.

Alternatively, a sensor may operate in passive mode to detect attacks and send alarms to the network security administrator when a network attack is taking place.

The sensor detects and prevents attacks with the use of six software modules: (1) an IP defragmentation module; (2) a flow manager software module; (3) a TCP reassembly software module; (4) a protocol anomaly detection module; (5) a stateful signature detection module; and (6) a traffic signature detection module.

The IP defragmentation software module reconstructs packets that were fragmented prior to reaching the sensor, that is, this module combines the packet fragments back into packets. After the packets are reconstructed, the flow manager software module organizes the packets into "packet flows" and associates them with a single communication session. That is, packets are organized according to whether they flow from a network client to the central management server or vice-versa, and according to whether they are part of a TELNET session, FTP session, HTTP session, and so on. In addition, the flow manager software module is capable of associating control and auxiliary flows within the same session. For example, FTP control flows and their associated FTP data flows are all combined in the same FTP session. The TCP packets in all the sessions are organized by the TCP reassembly software module, which orders the TCP packets that arrived out of order while removing packet overlaps and duplicate packets that were unnecessarily re-transmitted.

The IP defragmentation, flow manager, and TCP reassembly software modules enable the network intrusion detection and prevention sensor to search for security attacks faster and more accurately than other currently available network intrusion detection systems.

Intruding packets are detected and prevented from spreading to the private or outside networks by the protocol anomaly detection, stateful signature detection, and traffic signature detection software modules. Intruding packets are those containing network attack identifiers associated with network security breaches. Such network attack identifiers may be protocol irregularities, attack signatures, traffic signatures, or a combination of one or more of these, among others. The protocol anomaly detection module looks at the packet flows arranged by the flow manager software module to determine irregularities in the network protocol specifications in the packet. The stateful signature detection module matches known attack signatures to the TCP data stream in case of TCP packets and to the headers and data of packets transmitted with other network protocols. The traffic signature module matches traffic signatures

to the network traffic to detect attacks such as port scans and network sweeps. Incoming packets that are judged malicious are dropped by the sensor before reaching any of the network hosts and likewise, outgoing packets are dropped by the sensor before reaching the outside network. The sensor may also drop all the packets in a given session if one or more of its packets are considered to be malicious.

The sensor is also equipped with an IP router software module and an IP forwarder software module to route incoming and outgoing packets to the appropriate points in the network (IP router software module) and to use the routing information to forward the packets to their destination (IP forwarder software module.) The IP forwarder software module has full control over which packets will be allowed through the sensor and will not let packets that any of the other software modules has deemed malicious to go through.

The network intrusion detection and prevention central management server controls all the multiple sensors placed on the network using a single network security policy specified by the network security administrators. The security policy defines which traffic to inspect and which attacks the sensor should look for. The server validates the security policy, loads the security policy to all the sensors, maintains a history of policy changes, and collects the logs and alarms from the sensors for storage, display, and notification, among other functions. The server also keeps a central database to store the network security policy, including older and updated versions of the policy, attack signatures, logs and alarms, and other reporting information.

Network security administrators may view the logs and alarms by means of a network detection and prevention graphical user interface. The user interface can be accessed from any client connected to the network and provides access to all the management server and sensor functionalities. The user interface enables network security administrators to view information coming from the sensors and the server to determine what is happening in the network. The information provided by the sensors and the server is organized in reports that provide access to network statistics that otherwise would be difficult to gather, such as the top IP addresses used in attacks, the top attacks, the number of alarms and incidents generated, and whether an alarm is real or false, among other statistics. In addition, network security administrators use the user interface to define the network security policy and to instruct the central management server to distribute the security policy to some or all of the sensors. All

communications between the user interface, the server, and the sensors are protected by encryption and authentication mechanisms.

In general, in one aspect, the invention provides methods and apparatus, implementing and using techniques for preventing security breaches during an FTP connection. A network intrusion and detection sensor reconstructs packet fragments, organizes the packet fragments into an FTP packet flow, and reassembles a plurality of TCP fragments into a TCP stream. A software module inspects the TCP stream for protocol anomalies by determining whether the FTP packet flow is part of an FTP port command. If the software module determines that the FTP packet flow is part of an FTP port command, then the sensor compares an IP address of a user to an IP address of any port command wherever found in the TCP stream. If the sensor determines that the IP address of the user does not match the IP address associated with the port command, then the sensor drops the FTP packet flow and closes the FTP connection.

In general, another aspect, the invention provides methods and apparatus, implementing and using techniques for preventing security breaches during an SMTP connection. A network intrusion and detection sensor reconstructs packets fragments sent by a user, organizes the packet fragments into an SMTP packet flow, and reassembles a plurality of TCP packet fragments into a TCP stream. A software module determines whether there is an SMTP command present in the TCP stream, and, if so, the sensor searches for an attack signature in the SMTP command. If the sensor detects an attack signature in the SMTP command, then the sensor drops the SMTP packet flow and closes the SMTP connection.

Particular implementations can include one or more of the following features.

The attack signature can be a wiz command.

Advantageously, the systems and methods of the present invention detect and prevent network security breaches accurately and immediately. Those systems and methods are able to detect with low false alarm rates a multitude of attacks not detected by current network security products. In addition, the systems and methods of the present invention permit convenient, useful, and cost effective central management of an organization's network security.

Brief Description of the Drawings

The foregoing and other objects of the present invention will be apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which like reference characters refer to like parts throughout, and in which:

FIG. 1 is a schematic diagram of a prior art network environment protected by a firewall and a network intrusion detection system;

FIG. 2 is a schematic diagram of the software and hardware components used in the disclosed example of an MMIDP system;

FIG. 3 is a schematic diagram of a preferred MMIDP system and the network environment in which the systems and methods of the present invention operate;

FIG. 4 is a schematic diagram of an alternative MMIDP system and the network environment in which the systems and methods of the present invention operate;

FIG. 5 is a schematic diagram of another alternative MMIDP system and the network environment in which the systems and methods of the present invention operate;

FIG. 6 is a schematic view of the exemplary software modules used in the network intrusion detection and prevention sensor;

FIG. 7 is an exemplary flow table constructed by the flow manager software module;

FIG. 8 is a flow chart showing exemplary steps taken by the flow manager software module when new packets arrive at the network intrusion detection and prevention sensor;

FIG. 9 is a flow chart showing exemplary steps taken by the protocol anomaly detection software module when packets arrive at the network intrusion detection and prevention sensor running at gateway mode;

FIG. 10 is an exemplary table of protocols supported by the private network;

FIG. 11 is a flow chart showing exemplary steps taken by the stateful signature detection software module when packets arrive at the network intrusion detection and prevention sensor running at gateway mode;

FIG. 12 is a flow chart showing exemplary steps taken by the traffic signature detection software module when packets arrive at the network intrusion detection and prevention sensor running at gateway mode;

FIG. 13 is a flow chart showing exemplary steps taken by the network intrusion detection and prevention sensor when determining the validity of an incoming or outgoing packet;

FIG. 14 is a schematic view of exemplary functions performed by the network intrusion detection and prevention graphical user interface;

FIG. 15 is a schematic view of exemplary functions performed by the network intrusion detection and prevention central management server;

FIG. 16 is a flow chart illustrating exemplary steps taken by a network intrusion detection and prevention sensor, server, and graphical user interface when an FTP bounce attack is imminent on the network; and

FIG. 17 is a flow chart illustrating exemplary steps taken by a network intrusion detection and prevention sensor, server, and graphical user interface when a SMTP "wiz" attack is imminent on the network.

Detailed Description of the Invention

Referring to FIG. 1, a schematic diagram of a prior art network environment protected by a firewall and a network intrusion detection system is described. The connection between Internet 19 and private network 17, consisting of servers 16a and 16c and computers 16b and 16d, is guarded by firewall 18. Firewall 18 inspects all the packets flowing from Internet 19 to private network 17 and controls the access of users in private network 17 to outside resources. Any packet not conforming to static heuristics predetermined by the network access policy will be rejected by firewall 18, and not allowed inside private network 17.

Network intrusion detection system (NIDS) 20 is placed behind firewall 18 to inspect the packets allowed into network 17 by firewall 18. NIDS 20 is a passive device, capable only of sending an alarm to the network security administrator of private network 17 to warn that private network 17 is under attack, or in certain cases, of directing firewall 18 to take action if an attack is detected.

Referring now to FIG. 2, a schematic diagram of the software and hardware components used in the disclosed example of an MMIDP system is described. MMIDP system 23 is installed on a private network to detect and prevent security breaches on the network. MMIDP system 23 consists of MMIDP sensors 25a-d, MMIDP central management server 30, MMIDP database 35, and MMIDP graphical user interfaces ("GUIs") 40a-d.

MMIDP sensors 25a-d are hardware appliances placed at multiple gateway points on a private network, that is, at any point in the network that acts as an entrance to other networks, such as the Internet. MMIDP sensors 25a-d are all centrally managed from MMIDP server 30. Network administrators use MMIDP GUIs 40a-d to define a network security policy and to instruct MMIDP central management server 30 to distribute the security policy to some or all of MMIDP sensors 25a-d. The network security policy defines which traffic to inspect and which attacks MMIDP sensors 25a-d should look for.

In a preferred embodiment, MMIDP sensors 25a-d operate in gateway mode to prevent attacks by dropping any suspicious packet before it reaches its intended recipient, either inside or outside the private network or by interrupting or closing the network connection generating the attacks. MMIDP sensors 25a-d operating in gateway mode not only detect network attacks but also prevent them from occurring. Alternatively, MMIDP sensors 25a-d may operate in passive mode to detect attacks and send alarms that are displayed in MMIDP GUIs 40a-d to the network security administrators when a network attack is taking place. The network security administrators then may decide on an appropriate course of action to control the network attack.

MMIDP sensors 25a-d are equipped with eight software modules described below that operate on the network packets to detect and prevent network security breaches: (1) an IP defragmentation software module; (2) a flow manager software module; (3) a TCP reassembly software module; (4) a protocol anomaly detection software module; (5) a stateful signature detection software module; (6) a traffic signature detection software module; (7) an IP router software module; and (8) an IP forwarder software module.

MMIDP sensors 25a-d are all centrally managed from MMIDP server 30. MMIDP server 30 validates the network security policy defined by the network security administrators using MMIDP GUIs 40a-d, which transmit the policy to server 30, loads the security policy to some or all MMIDP sensors 25a-d, maintains a history of policy changes, and collects the logs and alarms from MMIDP sensors 25a-d for storage, display, and notification, among other functions, as described in detail below. In addition, MMIDP server 30 keeps MMIDP database 35 to store the network security policy, including older and updated versions of the policy, attack signatures, logs and alarms, and other reporting information.

Network security administrators use MMIDP GUIs 40a-d to analyze how MMIDP sensors 25a-d are handling incoming and outgoing network packets. MMIDP GUIs 40a-d can be accessed from any client connected to the network and provide access to all the functionalities of MMIDP sensors 25a-d and MMIDP server 30. MMIDP GUIs 40a-d enable network security administrators to view information coming from MMIDP sensors 25a-d and MMIDP server 30 to determine what is happening in the network and to take any subsequent action if necessary. The information provided by MMIDP sensors 25a-d and MMIDP server 30 is organized in reports that provide access to network statistics that otherwise would be difficult to gather, such as the top IP addresses used in attacks, the top attacks, the number of alarms and incidents generated, and whether an alarm is real or false, among other statistics. In addition, network security administrators may specify which signatures from the set of signatures stored in MMIDP database 35 will be used to detect and prevent attacks, as well as create new signatures. All communications between MMIDP sensors 25a-d, MMIDP server 30, MMIDP database 35, and MMIDP GUIs 40a-d are protected by encryption and authentication mechanisms.

Referring now to FIG. 3 is a schematic diagram of a preferred MMIDP system and the network environment in which the systems and methods of the present invention operate is described. MMIDP sensors 45a-c are placed at the gateway points of a private network consisting of remote office local area network 50, demilitarized zone ("DMZ") 55, and local area network 60, formed by wired network 65 and wireless network 70.

Wired network 65 is a local area network inside local area network 60 connecting MMIDP GUI 110a, personal computer user 67b, and notebook user 67c. Wireless network 70 is a wireless local area network inside local area network 60 connecting PDA user 73a and wireless telephone user 73b by means of base station 72. DMZ 55 is a neutral zone in the private network consisting of mail server 75 and web server 80 to handle all mail and web access requests from internal users in the network as well as from users outside of the network. DMZ 55 is used as a further layer of security to prevent outside users to have access to other servers in the private network besides mail server 75 and web server 80. It should be understood by one skilled in the art that remote office local area network 50, local area network 60, and DMZ 55 may comprise any electronic device capable of connecting to the Internet or other network operating with common protocols via a wired or wireless network, such

as personal computers, notebook computers, personal digital assistants, wireless telephone systems, and video game systems, among others.

MMIDP sensors 45a-c are positioned at multiple gateway points of the private network inside firewalls 85a-b to inspect all the incoming packets to the private network that were deemed secure by firewalls 85a-b as well as all outgoing packets that are not checked by firewalls 85a-b. Placing MMIDP sensors 45a-c inside firewalls 85a-b reduces the traffic that MMIDP sensors 45a-c need to analyze since only the packet flows and connections accepted by firewalls 85a-b need to be checked. In addition, placing MMIDP sensors 45a-c inside firewalls 85a-b allows network security administrators to evaluate the performance of firewalls 85a-b. Firewalls 85a-b may be packet filtering firewalls, circuit-level firewalls, application-level firewalls, or stateful inspection firewalls. Preferably, firewalls 85a-b are stateful inspection firewalls that serve as entrance points to Internet 90, with firewall 85b connected to router 95 for routing the incoming network packets to either DMZ 55 or local area network 60.

MMIDP server 100 in local area network 60 is able to centrally manage MMIDP sensors 45a-c. MMIDP server 100 also maintains MMIDP database 105 to store network security policies, attack signatures, logs and alarms, and other reporting information.

Network security administrators use MMIDP GUIs 110a-c to define a network security policy and to instruct MMIDP central management server 100 to distribute the security policy to some or all of MMIDP sensors 45a-c. The network security policy defines which traffic to inspect and which attacks MMIDP sensors 45a-c should look for. MMIDP GUIs 110a-c enable network security administrators to view information coming from MMIDP sensors 45a-c, and MMIDP server 100 to determine what is happening in the network formed by remote office local area network 50, DMZ 55, and local area network 60. The information provided by MMIDP sensors 45a-c and MMIDP server 100 is organized in reports that provide access to a list of all the detected attacks and intrusions as well as network statistics that otherwise would be difficult to gather, such as the top IP addresses used in attacks, the top attacks, the number of alarms and incidents generated, and whether an alarm is real or false, among other statistics. In addition, network security administrators may specify which signatures from the set of signatures stored in MMIDP database 105 will be used to detect and prevent attacks, as well as create new

signatures. It should be understood by one skilled in the art that MMIDP GUIs 110a-c are networking clients that may be placed on any network that has access to MMIDP server 100 through Internet 90.

Referring now to FIG. 4, a schematic diagram of an alternative MMIDP system and the network environment in which the systems and methods of the present invention operate is described. In this alternative, MMIDP sensors 45a-b are placed outside of firewalls 85a-c so that MMIDP sensors 45a-b are the entrance points to Internet 90. In addition, MMIDP sensor 45b is capable of supporting more than one network interface, such as network connection 47a and network connection 47b. This alternative may be used in cases where network security administrators are mostly concerned about attacks from outsiders. Placing MMIDP sensors 45a-b outside of firewalls 85a-c enables network security administrators to watch all the traffic that would typically be blocked by the firewall and would be undetected by an internal system.

Referring now to FIG. 5, a schematic diagram of another alternative MMIDP system and the network environment in which the systems and methods of the present invention operate is described. In this alternative, MMIDP sensors 45a-b are fully responsible for the security of the private network formed by remote office local area network 50, DMZ 55, and local area network 60. There are no firewalls being used to protect the private network. MMIDP sensors 45a-b analyze all the incoming and outgoing packets in the private network. This alternative may be used in cases where network security administrators are confident that MMIDP sensors 45a-b will be able to handle the volume of traffic to and from the network or in cases where network security administrators are not able to invest the time and money required to purchase an additional firewall system that has to be integrated and fully compliant with the other systems on the network.

Referring now to FIG. 6, a schematic view of the exemplary software modules used in the network intrusion detection and prevention sensor is described. MMIDP sensors 25a-d detect and prevent network security attacks with the use of eight software modules: (1) IP defragmentation software module 115; (2) flow manager software module 120; (3) TCP reassembly software module 125; (4) protocol anomaly detection software module 130; (5) stateful signature detection software module 135; (6) traffic signature detection software module 140; (7) IP router software module 145; and (8) IP forwarder software module 150.

IP defragmentation software module 115 reconstructs packets that were fragmented prior to reaching MMIDP sensors 25a-d. Packets are fragmented at network gateways when they are larger than the maximum packet size allowed in the network. The packets are reassembled according to the algorithm specified in the RFC 815 standard of the Internet Engineering Task Force. The algorithm can reassemble any number of packet fragments arriving in any order with any possible pattern of fragment overlap and duplication by keeping a buffer of length equal to the length of the packet being reassembled. The length of the packet is specified in the packet header. IP defragmentation software module 115 also performs security verification checks on the packet fragments, throwing out and reporting fragments whose parameters (such as packet length or packet offset) are known to be malicious and potentially dangerous.

After the packets are reconstructed by IP defragmentation software module 115, flow manager software module 120 organizes the packets into "packet flows", also referred to as flows, and associates them with a single communication session. A packet flow is a sequence of packets that flow from a source to a destination. That is, packets are organized according to whether they originate at the private network and flow to the outside network or vice-versa, and according to whether they are part of a TELNET session, FTP session, HTTP session, and so on. Control and data flows are grouped into the same session. Flow manager software module 120 organizes all the packet flows coming to and from the private network into a flow table that is implemented as a hash table for easy access by software modules 130, 135, and 140.

Referring now to FIG. 7, an exemplary flow table constructed by the flow manager software module is described. Flow table 155 is implemented as a hash table that organizes the packets coming into MMIDP sensors 25a-d into packet flows and sessions. The hash table may have "n" cells or buckets, such as the 8 hash buckets shown for flow table 155. Each bucket in the table consists of a pointer to a linked list of packet flow descriptors that is addressed by a hash value. The hash value is computed by a perfect hash function that hashes the values of a 5-tuple consisting of <source IP address, source port, destination IP address, destination port, protocol> into a unique integer in the range of 1 to "n". For example, flow table 155 contains hash table buckets 153a-h, with each bucket being addressed by an integer hash value ranging from 1 to 8. Furthermore, each packet flow descriptor is addressed by a 5-

tuple key which is unique to that flow and is made of that flow's 5-tuple <source IP address, source port, destination IP address, destination port, protocol>.

The packet flow descriptors addressed by each key consist of information about each specific packet flow, including the 5-tuple above as well as the list of packets that belong to the described packet flow. For example, hash table bucket 153a points to packet flow descriptors 156a and 156b, while hash table bucket 153c points to packet flow descriptor 157. In addition, each packet flow in the list is associated to a session, such as TELNET session 161, FTP session 162, and HTTP session 163. The association is done by a double pointer (represented by the double arrows in FIG. 7) so that each packet flow descriptor points to a session and the session points back to each packet flow descriptor. The double pointer enables protocol anomaly detection software module 130, stateful signature detection software module 135, and traffic signature detection software module 140 to quickly and accurately retrieve information about incoming packet flows and their associated sessions. Packet flow descriptor 156a, for example, contains information about a TELNET flow from source A to destination B, as well as a list of packets that belong to that packet flow. Packet flow descriptors addressed by the same hash key (and belonging to the same hash bucket) may point to different sessions and packet flows descriptors belonging to the same session may be addressed by different hash keys. For example, packet flow descriptors 156a-b are both in hash bucket 153a, but packet flow descriptor 156a is associated to TELNET session 161 while packet flow descriptor 156b is associated to FTP session 162, which is also associated to packet flow descriptors 157, 158, and 159b, all belonging to different hash buckets.

Referring now to FIG. 8, a flow chart showing exemplary steps taken by the flow manager software module when new packets arrive at the network intrusion detection and prevention sensor is described. When a new flow of packets arrive at MMIDP sensors 25a-d, flow manager software module 120 identifies the source, destination, the source port, the destination port, and the protocol used for the packets at step 170 to compute the perfect hash function that maps the 5-tuple identifier into a distinct integer key at step 175. At step 180, flow manager software module 120 determines whether the key addresses an already existing packet flow descriptor in the hash table. If the key does not correspond to an existing packet flow descriptor, a new packet flow descriptor is inserted in the table at step 185.

At step 190, the system extracts a pointer to the packet flow descriptor for the incoming packet. Lastly, at step 200, flow manager software module 120 passes the pointer to the packet flow descriptor and its corresponding session, as extracted in step 190, to detection modules 130, 135, and 140. This enables protocol anomaly detection software module 130, stateful signature detection module 135, and traffic signature detection software module 140 to quickly and accurately retrieve information about an incoming packet flow descriptor and its associated session from its pointer.

Referring back to FIG. 6, the TCP packets in all the packet flows in the flow table are reassembled by TCP reassembly software module 125. TCP reassembly software module 125 arranges TCP packets that are part of a stream of packets in their correct order, while removing duplicate packets and packet overlap. Each TCP packet has a sequence number in its header, which enables software module 125 to rearrange TCP packets in their correct order when they arrive out of sequence or when they are unnecessarily re-transmitted in case they are delayed in the network by a longer time period than tolerated by the network.

IP defragmentation software module 115, flow manager software module 120, and TCP reassembly software module 125 enable MMIDP sensors 25a-d to detect and prevent security attacks faster and more accurately than other currently available intrusion detection systems. Intruding packets are detected and prevented from spreading to the private or outside networks by protocol anomaly detection software module 130, stateful signature detection software module 135, and traffic signature detection software module 140.

Referring now to FIG. 9, a flow chart showing exemplary steps taken by the protocol anomaly detection software module when packets arrive at the network intrusion detection and prevention sensor running at gateway mode is described. Protocol anomaly detection software module 130 examines the packet flows arranged by flow manager software module 125 in flow table 155 to determine irregularities in the network protocol specifications in non-TCP packets and TCP data streams. At step 215, protocol anomaly detection software module 130 accesses the packet flow descriptor and session corresponding to the packets arriving at MMIDP sensors 25a-d from the pointer to the packet flow descriptor and session passed by flow manager software module 120.

At step 220, protocol anomaly detection software module 130 examines the packet flow and session to determine which protocols need to be checked for irregularities. At step 225, protocol anomaly detection software module 130 performs high speed protocol verification by querying a protocol database that contains a list of protocols supported by MMIDP system 23 and the allowable actions for each protocol. Protocol anomaly detection software module 130 queries the protocol database to determine whether the incoming packets are compliant with the protocol used to transmit them and whether the actions or commands embodied in the non-TCP packets or TCP data streams are authorized or allowed for the protocol. Steps 220 and 225 enable protocol anomaly detection 130 to check for protocol irregularities much more quickly and accurately than other typical anomaly detection systems. If the protocol specifications in the protocol database do not match the protocol specifications in the packets (step 230), the packets are dropped at step 235.

Referring now to FIG. 10, an exemplary table of protocols supported by the private network is described. Protocol table 245 lists the protocols that are supported by MMIDP system 23 and some of their corresponding RFC standard specification, if any. It should be understood by one skilled in the art that additional protocols not listed in protocol table 245 such as ICMP may also be supported by the private network.

Referring back to FIG. 6, stateful signature detection software module 135 matches known attack signatures to the packet headers and data according to the network protocol used to transmit the packet. Software module 135 downloads known attack signatures from MMIDP database 35 run by MMIDP server 30 each time a signature update is made. Signature updates are made whenever new signature attack patterns are learned by network security administrators or by the vendors of MMIDP system 23. Preferably, new signatures will be updated within a week of being characterized in the Internet or other public domain forums.

The signatures are compared only to the relevant portion of the data stream or data packets. This is done utilizing two mechanisms. The first makes sure that signatures are only compared against traffic from relevant packet flows. For example, SMTP signatures will not be compared against FTP data. The second mechanism analyzes the traffic to understand the state of the packet and data stream communications. This analysis allows MMIDP to distinguish, for example, between SMTP commands and SMTP data lines or FTP user names and FTP file names. That

is, stateful signature detection software module 135 compares signatures that are relevant to the data protocol to the relevant portion of the data. For example, not only will a signature that looks for a certain SMTP command be compared only to SMTP traffic, but the comparison is restricted to what is analyzed to be an SMTP command in the SMTP traffic. That is, by examining the attributes of the packet flow's session entry, such as sessions 161-163 in flow table 155, signature detection software module 135 is able to determine which signatures need to be matched against the packet flow. This considerably improves the performance of signature detection software module 135 since only the signatures that are meaningful to the packet flow need to be analyzed.

Referring now to FIG. 11, a flow chart showing exemplary steps taken by the stateful signature detection software module when packets arrive at the network intrusion detection and prevention sensor running at gateway mode is described. At step 210, stateful signature detection software module 135 accesses the packet flow descriptor and session corresponding to the packets arriving at MMIDP sensors 25a-d from the pointer to the packet flow descriptor and session passed by flow manager software module 120.

At step 260, software module 135 queries MMIDP database 35 to find the signatures that are relevant to the incoming data stream or packets. The relevant signatures are those that would only be considered attacks in the context of the packet flow and session retrieved from the flow table. The relevant signatures are converted into regular expressions when stored in database 35. Regular expressions are patterns that describe portions of strings. For example, the regular expression "[0123456789]" matches any single digit in UNIX-based operating systems. Converting the signatures into regular expressions enables software module 135 to efficiently match signatures against packets.

At step 265, software module 135 checks whether the incoming packets belong to a TCP flow. If not, at step 275, the signatures are compared to the incoming packets using, for example, Deterministic Finite Automata ("DFA"). DFA signature matching builds a state machine for each regular expression to quickly decide whether the regular expression is present in the incoming packets. If the incoming packets are of a TCP flow, the signatures are compared to the entire TCP data stream (step 270).

If any matching signatures are found (step 280), the corresponding packets and the flow to which they belong are dropped by software module 135 at step 290.

Otherwise, the incoming packets free of matching signatures are delivered to traffic signature detection software module 140. It is understood by those skilled in the art that other pattern matching algorithms besides DFA matching can be used to match attack signatures.

Referring back to FIG. 6, traffic signature software module 140 matches traffic signatures to the network traffic to detect, for example, port scans and network sweeps. The traffic signatures are downloaded to software module 140 from MMIDP database 35 maintained by MMIDP server 30.

Referring now to FIG. 12, a flow chart showing exemplary steps taken by the traffic signature detection software module when packets arrive at the network intrusion detection and prevention sensor running at gateway mode is described. The steps taken by traffic signature detection software module 140 are similar to those taken by stateful signature detection software module 135 to detect attack signatures. At step 310, traffic signature detection software module 140 accesses the packet flow descriptor and session corresponding to the packets arriving at MMIDP sensors 25a-d from the pointer to the packet flow descriptor and session passed by flow manager software module 120.

At step 315, traffic signature detection software module 140 queries MMIDP database 35 to find the traffic signatures that are relevant to the flow of the incoming packets. The relevant signatures are found by examining the protocol of the flow to which the incoming packets belong. For example, if the incoming packets are part of an ICMP packet flow, software module 140 will only consider ICMP-based traffic signatures.

At step 320, the traffic signatures are matched to the incoming data stream or packets. If any matching signatures are found, software module 140 updates a signature-specific count as specified by the traffic signature at step 325. The signature count may, for example, count how many different hosts were contacted from the same IP address, during a given time period, and so on. If the signature count is above a pre-determined threshold (step 330), then software module 140 generates an alarm to be displayed at MMIDP GUIs 40a-d at step 335.

Referring back to FIG. 6, MMIDP sensors 25a-d are also equipped with IP router software module 145 and IP forwarder software module 150 to route incoming and outgoing packets to the appropriate points in the network (IP router software module 145) and to use the routing information to forward the packets to their

destination (IP forwarder software module 150). IP forwarder software module 150 has full control over which packets will be allowed through MMIDP sensors 25a-d and will not let packets that any of the other software modules has deemed malicious to go through.

Referring now to FIG. 13, a flow chart showing exemplary steps taken by the network intrusion detection and prevention sensor when determining the validity of an incoming or outgoing packet is described. At step 350, the packet fragments arriving at MMIDP sensors 25a-d are reconstructed into packets by IP defragmentation software module 115. At step 355, flow manager software module 120 in MMIDP sensors 25a-d organizes the incoming packets into packet flows and sessions in a flow table as described above. At step 360, MMIDP sensors 25a-d check whether there are any TCP packets among the incoming packets. If so, the TCP packets are reordered at step 365. At step 370, protocol anomaly detection software module 130 checks to see if there are any protocol irregularities in the packets. Any packet presenting protocol irregularities will be dropped at step 380.

The packets conforming to the network protocol specifications of the protocols listed in table 245 (FIG. 10) will then proceed to stateful signature detection software module 135 at step 375 to be matched against attack signatures downloaded to MMIDP sensors 25a-d from MMIDP database 35. As described above, only the relevant signatures are checked, thereby considerably speeding up the signature matching process as compared to previously-known signature-based systems. If there are any signatures matching information in a given non-TCP packet or TCP data stream, the packet or stream is dropped at step 380.

Packets containing no matching signatures are passed on to traffic signature detection software module 140 at step 385 for determining whether there are any traffic signatures that match the packet flows associated with the packets being analyzed. If there are any matching traffic signatures and the internal counters of any of these traffic signatures surpasses a pre-determined threshold (steps 390, 400), then MMIDP sensors 25a-d generate an alarm at step 405 to be displayed at MMIDP GUIs 40a-d indicating a network sweep or port scan at the network.

Lastly, all the packets free of protocol irregularities and matching attack and traffic signatures are routed and forwarded to their appropriate network destinations by IP router software module 145 and IP forwarder software module 150 at step 410. It should be understood by one skilled in the art that all the steps described above in

connection with FIG. 13 are performed upon the arrival of each new packet at MMIDP sensors 25a-d. It should also be understood by one skilled in the art that steps 370, 375, and 385 may be performed in a different order.

Referring now to FIG. 14, a schematic view of exemplary functions performed by the network intrusion detection and prevention graphical user interface is described. MMIDP GUIs 40a-d can be accessed from any client connected to the network and provide access to all the functionalities of MMIDP server 30 and MMIDP sensors 25a-d. Configuration interface 420 allows network security administrators to install MMIDP sensors 25a-d and perform other configuration functions related to their maintenance. Security policy editor 425 enables network security administrators to specify a network security policy to define which traffic to inspect and which attacks MMIDP sensors 25a-d should look for. Logs and alarms viewer 430 enables network security administrators to view information coming from MMIDP sensors 25a-d and MMIDP server 30 to determine what is happening in the network. Logs describe the packet activity coming through MMIDP sensors 25a-d and alarms are generated by MMIDP sensors 25a-d when an attack has been attempted on the network. The alarms are classified into new, real, false, or closed, that is, alarms that are no longer active due to the packets attempting the attack being dropped. Network security administrators may view logs according to the order in which they are generated by MMIDP sensors 25a-d and according to other specified criteria such as their date, the source IP address, the destination IP address, and so on. The logs may be viewed in real time and at different levels of detail. All the logs may be backed up and stored in MMIDP database 35.

The information provided by MMIDP sensors 25a-d and MMIDP server 30 is organized in reports that provide access to network statistics that otherwise would be difficult to gather, such as the top IP addresses used in attacks, the top attacks, the number of alarms and incidents generated, and whether an alarm is real or false, among other statistics. The reports are displayed within reports viewer 435. In addition, network security administrators may specify which signatures from the set of signatures stored in MMIDP database 35 will be used to detect and prevent attacks, as well as create new signatures.

Lastly, status viewer 440 enables network security administrators to monitor the status of MMIDP sensors 25a-d, MMIDP server 30, and other network resources.

It is understood by one skilled in the art that MMIDP GUIs 40a-d may perform additional functions other than the ones described above in connection with FIG. 14.

Referring now to FIG. 15, a schematic view of exemplary functions performed by the network intrusion detection and prevention central management server is described. MMIDP server 30 collects the logs and alarms from MMIDP sensors 25a-d (445) for storage, display, and notification, and information about the status of MMIDP sensors 25a-d (450), among other functions. In addition, MMIDP server 30 keeps MMIDP database 35 to store the network security policy (455), attack signatures, logs and alarms, and other reporting information. Whenever MMDIP sensors 25a-d match attack and traffic signatures against incoming and outgoing packets, MMIDP server 30 distributes the network security policy or policy updates stored in MMDIP database 35 to the sensors (460). MMIDP server 30 is also responsible for updating MMIDP database 35 whenever new signatures are specified by network security administrators using MMIDP GUIs 40a-d (465). It is understood by one skilled in the art that MMIDP server 30 may perform additional functions other than the ones described above in connection with FIG. 15.

Referring now to FIG. 16, a flow chart illustrating exemplary steps taken by a network intrusion detection and prevention sensor, server, and graphical user interface when an FTP bounce attack is imminent on the network is described. At step 475, a user connects to an FTP server in the network to download or upload files. When this happens, the FTP user's software provides the FTP server at step 480 an IP address and a port number to which the file should be sent or taken from. This is done via an FTP "port" command. In practice, the IP address is that of the user, but the port command does not limit the IP address to the user's address. Because of this, an attacker can tell the FTP server to open a connection to an IP address that is different from the user's address and transfer files from the FTP server to it. To detect this attack, the MMIDP sensor needs to compare the requests to the port command with the IP address of the user and send an alarm to the user or close the FTP connection if the IP addresses do not match.

At step 485, the user sends an IP address to the FTP server that is different from the user's IP address. Prior to the packets containing the user's IP address reach the FTP server, the MMIDP sensor reconstructs any packet fragments at step 490 and organizes the packets into an incoming FTP packet flow at step 495. At step 500, the MMID sensor reassembles the TCP packet fragments into client-to-server and server-

to-client data streams. At step 505, protocol anomaly detection software module 130 in the MMIDP sensor checks whether the packet is part of an FTP port command. If it is, the MMIDP sensor compares the IP address of the user to the one specified by the port command. At step 510, MMIDP checks if there was no PORT command, or if the IP address match. If either is true, the MMIDP sensor skips to step 520. If there was a PORT command and the IP address did not match, the MMIDP sensor drops the corresponding FTP packets, sends an alarm to MMIDP server 30, and closes the FTP connection at step 515. Lastly, at step 520, MMIDP server 30 collects log and packet information from the MMIDP sensor and sends it to MMIDP GUIs 40a-d for display.

Referring now to FIG. 17, a flow chart illustrating exemplary steps taken by a network intrusion detection and prevention sensor, server, and graphical user interface when an SMTP "wiz" attack is imminent on the network is described. The "wiz" attack occurs when an attacker uses the "wiz" command in an SMTP session with certain vulnerable SMTP servers to unlawfully gain root access on a network host. When successful, the attacker can take complete control over the network host, use it as a platform for launching further attacks, steal e-mail and other data, and ultimately gain access to the network resources. Since the "wiz" string can often appear in an e-mail body, recipient list, and so on, there is a high probability of generating false alarms if the signature matching is not done within the context of a client to server SMTP flow in "command mode."

At step 535, a user connects to an SMTP server in the network to establish an SMTP session. At step 540, the SMTP server establishes a TCP connection with the user through a 3-way handshake by exchanging SYN and ACK packets. When the TCP connection is established, the user sends the "wiz" command to the SMTP mail server at step 545 while the sendmail session is in command mode. At step 550, the MMIDP sensor reconstructs any packet fragments sent by the user. The reconstructed packets are organized into a SMTP packet flow at step 555. At step 560, the MMIDP sensor reassembles the TCP packet fragments into client-to-server and server-to-client data streams.

If there is an SMTP command present in the client-to-server data stream (step 565), the MMIDP sensor searches for the "wiz" signature in the SMTP command(s) at step 570. Once a signature match is found, the MMIDP sensor drops the SMTP packets, sends an alarm to MMIDP server 30, and closes the SMTP connection at step

575. Lastly, at step 580, MMIDP server 30 collects log and packet information from the MMIDP sensor and sends it to MMIDP GUIs 40a-d for display.

Although particular embodiments of the present invention have been described above in detail, it will be understood that this description is merely for purposes of illustration. Specific features of the invention are shown in some drawings and not in others, for purposes of convenience only, and any feature may be combined with other features in accordance with the invention. Steps of the described processes may be reordered or combined, and other steps may be included. Further variations will be apparent to one skilled in the art in light of this disclosure and such variations are intended to fall within the scope of the appended claims.

What is Claimed Is:

1. A method for detecting and preventing security breaches in network traffic, the method comprising:

reassembling a plurality of TCP packets in the network traffic into a TCP stream;

inspecting the TCP stream to detect information indicative of security breaches;

dropping a TCP packet from the TCP stream if the TCP stream contains information indicative of security breaches; and

forwarding a TCP packet from the TCP stream to a network destination if the TCP stream does not contain information indicative of security breaches.

2. The method of claim 1, wherein inspecting the TCP stream to detect information indicative of security breaches comprises inspecting the TCP stream for protocol irregularities.

3. The method of claim 1, wherein inspecting the TCP stream to detect information indicative of security breaches comprises searching the TCP stream for attack signatures.

4. The method of claim 3, wherein searching the TCP stream for attack signatures comprises using stateful signature detection.

5. The method of claim 1, wherein inspecting the TCP stream to detect information indicative of security breaches comprises using a plurality of network intrusion detection methods.

6. The method of claim 5, wherein the plurality of network intrusion detection methods comprises at least protocol anomaly detection.

7. The method of claim 5, wherein the plurality of network intrusion detection methods comprises at least signature detection.

8. The method of claim 1, further comprising grouping the plurality of TCP packets into packet flows and sessions.

9. The method of claim 1, further comprising storing the packet flows in packet flow descriptors.

10. The method of claim 9, further comprising searching the packet flow descriptors for traffic signatures.

11. The method of claim 9, wherein inspecting the TCP stream comprises searching for a network attack identifier in the TCP stream based on the packet flow descriptors and sessions associated with the TCP stream.

12. The method of claim 11, wherein the network attack identifier comprises a protocol irregularity.

13. The method of claim 11, wherein the network attack identifier comprises an attack signature.

14. The method of claim 11, wherein the network attack identifier comprises a plurality of network attack identifiers.

15. The method of claim 14, wherein the plurality of network attack identifiers comprises at least a protocol irregularity.

16. The method of claim 14, wherein the plurality of network attack identifiers comprises at least an attack signature.

17. The method of claim 13, wherein the attack signatures and the traffic signatures are stored in a signatures database.

18. The method of claim 8, wherein grouping the plurality of TCP packets into packet flows and sessions comprises storing the packet flows and sessions in a hash table.

19. The method of claim 18, wherein storing the packet flows and sessions in a hash table comprises computing a hash value from a 5-tuple consisting of: a source IP address; a destination IP address; a source port; a destination port; and a protocol type.

20. The method of claim 2, wherein inspecting the TCP stream for protocol irregularities comprises:

storing a plurality of protocol specifications supported by the network in a protocol database; and

querying the protocol database to determine whether the plurality of TCP packets in the packet flows and sessions are compliant with one or more of the plurality of protocol specifications in the protocol database.

21. The method of claim 3, wherein searching the TCP stream for attack signatures comprises querying the signatures database to determine whether there are matching signatures in the TCP stream.

22. The method of claim 21, wherein determining whether there are matching signatures in the TCP stream comprises using DFA for pattern matching.

23. The method of claim 1, further comprising reconstructing the plurality of TCP packets from a plurality of packet fragments.

24. A system for detecting and preventing security breaches in network traffic, the system comprising:

a TCP reassembly software module for reassembling a plurality of TCP packets in the network traffic into a TCP stream;

a software module for inspecting the TCP stream to detect information indicative of security breaches;

a software module for dropping a TCP packet from the TCP stream if the TCP stream contains information indicative of security breaches; and

a software module for forwarding a TCP packet from the TCP stream to a network destination if the TCP stream does not contain information indicative of security breaches.

25. The system of claim 24, further comprising an IP defragmentation software module for reconstructing a plurality of packet fragments into the plurality of TCP packets.

26. The system of claim 24, further comprising a flow manager software module for grouping the plurality of TCP packets into packet flows and sessions.

27. The system of claim 26, wherein the flow manager software module comprises a routine for storing the packet flows and sessions into a hash table.

28. The system of claim 27, wherein the routine for storing the packet flows and sessions into a hash table comprises a routine for storing the packet flows in packet flow descriptors.

29. The system of claim 27, wherein the routine for storing the packet flows and sessions into a hash table comprises a routine for computing a hash value from a 5-tuple consisting of: a source IP address; a destination IP address; a source port; a destination port; and a protocol type.

30. The system of claim 24, wherein the software module for inspecting the TCP stream to detect information indicative of security breaches comprises a protocol anomaly detection software module.

31. The system of claim 24, wherein the software module for inspecting the TCP stream to detect information indicative of security breaches comprises a stateful signature detection software module.

32. The system of claim 28, further comprising a traffic signature detection software module for searching the packet flow descriptors for traffic signatures.

33. The system of claim 24, wherein the software module for inspecting the TCP stream for information indicative of security breaches comprises a plurality of software modules.

34. The system of claim 33, wherein the pluralify of software modules comprises at least a protocol anomaly detection software module.

35. The system of claim 33, wherein the plurality of software modules comprises at least a stateful signature detection software module.

36. The system of claim 34, wherein the protocol anomaly detection software module comprises:

a routine for storing a plurality of protocol specifications supported by the network in a protocol database; and

a routine for querying the protocol database to determine whether the plurality of TCP packets in the packet flows and sessions are compliant with one or more of the plurality of protocol specifications in the protocol database.

37. The system of claim 36, wherein the protocol specifications comprise specifications of one or more of: TCP protocol; HTTP protocol; SMTP protocol; FTP protocol; NETBIOS protocol; IMAP protocol; POP3 protocol; TELNET protocol; IRC protocol; RSH protocol; REXEC protocol; and RCMD protocol.

38. The system of claim 35, wherein the stateful signature detection software module comprises a routine for querying a signatures database to determine whether there are matching attack signatures in the TCP stream.

39. The system of claim 38, wherein the routine comprises using DFA for pattern matching.

40. The system of claim 24, further comprising:

a routine for collecting a plurality of security logs and alarms recording information about security breaches found in the TCP stream;

a routine for storing a network security policy identifying the network traffic to inspect and a plurality of network attacks to be detected and prevented;

a routine for distributing the network security policy to one or more gateway points in the network; and

a routine for updating the protocol database and the signatures database.

41. The system of claim 24, further comprising a graphical user interface comprising:

a routine for displaying network security information to network security administrators; and

a routine for specifying a network security policy.

42. A system for detecting and preventing security breaches in network traffic, the system comprising:

a network intrusion detection and prevention sensor placed in a network gateway, wherein the network intrusion detection and prevention sensor comprises:

a routine for reassembling a plurality of TCP packets into a TCP stream;

a software module for inspecting the TCP stream to detect information indicative of security breaches;

a software module for dropping a TCP packet from the TCP stream if the TCP stream contains information indicative of security breaches; and

a software module for forwarding a TCP packet from the TCP stream to a network destination if the TCP stream does not contain information indicative of security breaches;

a central management server to control the network intrusion detection and prevention sensor; and

a graphical user interface for configuring the network intrusion detection and prevention sensor.

43. The system of claim 42, wherein the network intrusion detection and prevention sensor is placed inside a firewall.

44. The system of claim 42, wherein the network intrusion detection and prevention sensor is placed outside a firewall.

45. The system of claim 42, wherein the network intrusion detection and prevention sensor further comprises an IP defragmentation software module for reconstructing a plurality of packet fragments into the plurality of TCP packets.

46. The system of claim 42, wherein the network intrusion detection and prevention sensor further comprises an IP router software module for routing a TCP packet from the TCP stream if the TCP stream does not contain information indicative of network security breaches through the network.

47. The system of claim 42, wherein the network intrusion detection and prevention sensor further comprises a flow manager software module for grouping the plurality of packets into packet flows and sessions.

48. The system of claim 47, wherein the flow manager software module comprises a routine for storing the packet flows in packet flow descriptors.

49. The system of claim 42, wherein the software module for inspecting information indicative of security breaches comprises a protocol anomaly detection software module.

50. The system of claim 42, wherein the software module for inspecting information indicative of security breaches comprises a stateful signature detection software module.

51. The system of claim 48, further comprising a traffic signature detection software module for searching the packet flow descriptors for traffic signatures.

52. The system of claim 42, wherein the software module for inspecting information indicative of security breaches comprises a plurality of software modules.

53. The system of claim 52, wherein the plurality of software modules comprises at least a protocol anomaly detection software module.

54. The system of claim 52, wherein the plurality of software modules comprises at least a stateful signature detection software module.

55. The system of claim 42, wherein the central management server comprises:

a routine for collecting a plurality of security logs and alarms recording information about security breaches found in the TCP stream;

a routine for storing a network security policy identifying the network traffic to inspect and a plurality of network attacks to be detected and prevented; and

a routine for distributing the network security policy to the network intrusion detection and prevention sensor.

56. The system of claim 42, wherein the graphical user interface comprises:

a routine for displaying network security information to network security administrators;

a routine for displaying status information on the network intrusion detection and prevention sensor; and

a routine for specifying a network security policy.

57. A network intrusion detection and prevention sensor for detecting and preventing network security breaches at a network gateway, the network intrusion detection and prevention sensor comprising:

a flow manager software module for grouping a plurality of packets into packet flows and sessions;

a TCP reassembly software module for reassembling a plurality of TCP packets from the plurality of packets into a TCP stream;

a software module for inspecting the TCP stream according to the packet flows and sessions to detect information indicative of security breaches;

a software module for dropping a packet from the plurality of packets if the TCP stream contains information indicative of security breaches; and

a software module for forwarding a packet from the plurality of packets to a network destination if the TCP stream does not contain information indicative of security breaches.

58. The network intrusion detection and prevention sensor of claim 57, further comprising an IP defragmentation software module for reconstructing a plurality of packet fragments into the plurality of TCP packets.

59. The network intrusion detection and prevention sensor of claim 57, wherein the network intrusion detection and prevention sensor further comprises an IP router software module for routing a TCP packet from the TCP stream if the TCP stream does not contain information indicative of network security breaches through the network.

60. The network intrusion detection and prevention sensor of claim 57, wherein the network intrusion detection and prevention sensor is controlled by a network security policy specifying the network traffic to inspect and a plurality of network attacks to be detected and prevented.

61. The network intrusion detection and prevention sensor of claim 60, wherein the network security policy is defined by a network security administrator using a graphical user interface.

62. The network intrusion detection and prevention sensor of claim 57, wherein the graphical user interface comprises:

a routine for displaying network security information to network security administrators;

a routine for displaying status information on the network intrusion detection and prevention sensor; and

a routine for specifying the network security policy.

63. The network intrusion detection and prevention sensor of claim 60, wherein the security policy is stored and distributed to the network intrusion detection and prevention sensor by a central management server.

64. The network intrusion detection and prevention sensor of claim 63, wherein the central management server comprises a routine for collecting a plurality

of security logs and alarms recording information about security breaches found in the TCP stream.

65. The network intrusion detection and prevention sensor of claim 57, wherein the software module for inspecting the TCP stream according to the packet flows and sessions to detect information indicative of security breaches comprises a protocol anomaly detection software module.

66. The network intrusion detection and prevention sensor of claim 57, wherein the software module for inspecting the TCP stream according to the packet flows and sessions to detect information indicative of security breaches comprises a stateful signature detection software module.

67. The network intrusion detection and prevention sensor of claim 58, wherein the software module for inspecting the plurality of packets according to the packet flows and sessions to detect information indicative of security breaches comprises a plurality of software modules.

68. The network intrusion detection and prevention sensor of claim 67, wherein the plurality of software modules comprises at least a protocol anomaly detection software module.

69. The network intrusion detection and prevention sensor of claim 67, wherein the plurality of software modules comprises at least a stateful signature detection software module.

70. A method for preventing security breaches during an FTP connection, the method comprising:

reconstructing packet fragments and organizing the packet fragments into an FTP packet flow;

reassembling a plurality of TCP fragments into a TCP stream;

inspecting the TCP stream for protocol anomalies, including determining whether the FTP packet flow is part of an FTP port command and

comparing an IP address of a user to an IP address of any port command wherever found in the TCP stream;

if the FTP packet flow is part of an FTP port command and if the IP address of the user does not match the IP address associated with the port command, then:

dropping the FTP packet flow and closing the FTP connection.

71. A system for preventing security breaches during an FTP connection, the system comprising:

a network intrusion and detection sensor to:

reconstruct, prior to any FTP packets containing an IP address of a user reaching an FTP server, any packet fragments;

organize the packet fragments into a FTP packet flow; and

reassemble a plurality of TCP fragments into a TCP stream; and

a software module to inspect the TCP stream for protocol anomalies by determining whether the FTP packet flow is part of an FTP port command;

if the software module determines that the FTP packet flow is part of an FTP port command, then the sensor is configured to compare an IP address of a user to an IP address of any port command wherever found in the TCP stream;

if the sensor determines that the IP address of the user does not match the IP address associated with the port command, then the sensor is further configured to drop the FTP packet flow and close the FTP connection.

72. A method for preventing security breaches during an SMTP connection, the method comprising:

reconstructing packets fragments sent by a user;

organizing the packet fragments into an SMTP packet flow;

reassembling a plurality of TCP packet fragments into a TCP stream;

inspecting the TCP stream for an attack signature, including determining whether there is an SMTP command present in the TCP stream and searching for an attack signature in the SMTP command;

if there is an SMTP command present in the TCP stream and if an attack signature is found in the SMTP command, then:

dropping the SMTP packet flow and closing the SMTP connection.

73. The method of claim 72, wherein the attack signature is a wiz command.

74. A system for preventing security breaches during an SMTP connection, the system comprising:

a network intrusion and detection sensor to:

reconstruct packets fragments sent by a user;

organizing the packet fragments into an SMTP packet flow;

reassembling a plurality of TCP packet fragments into a TCP stream;

a software module to inspect the TCP stream for an attack signature by first determining whether there is an SMTP command present in the TCP stream;

if the software module determines that there is an SMTP command present in the TCP stream, then the sensor is configured to search for an attack signature in the SMTP command;

if the sensor detects an attack signature in the SMTP command, then the sensor is further configured to drop the SMTP packet flow and close the SMTP connection.

75. The system of claim 74, wherein the attack signature is a wiz command.

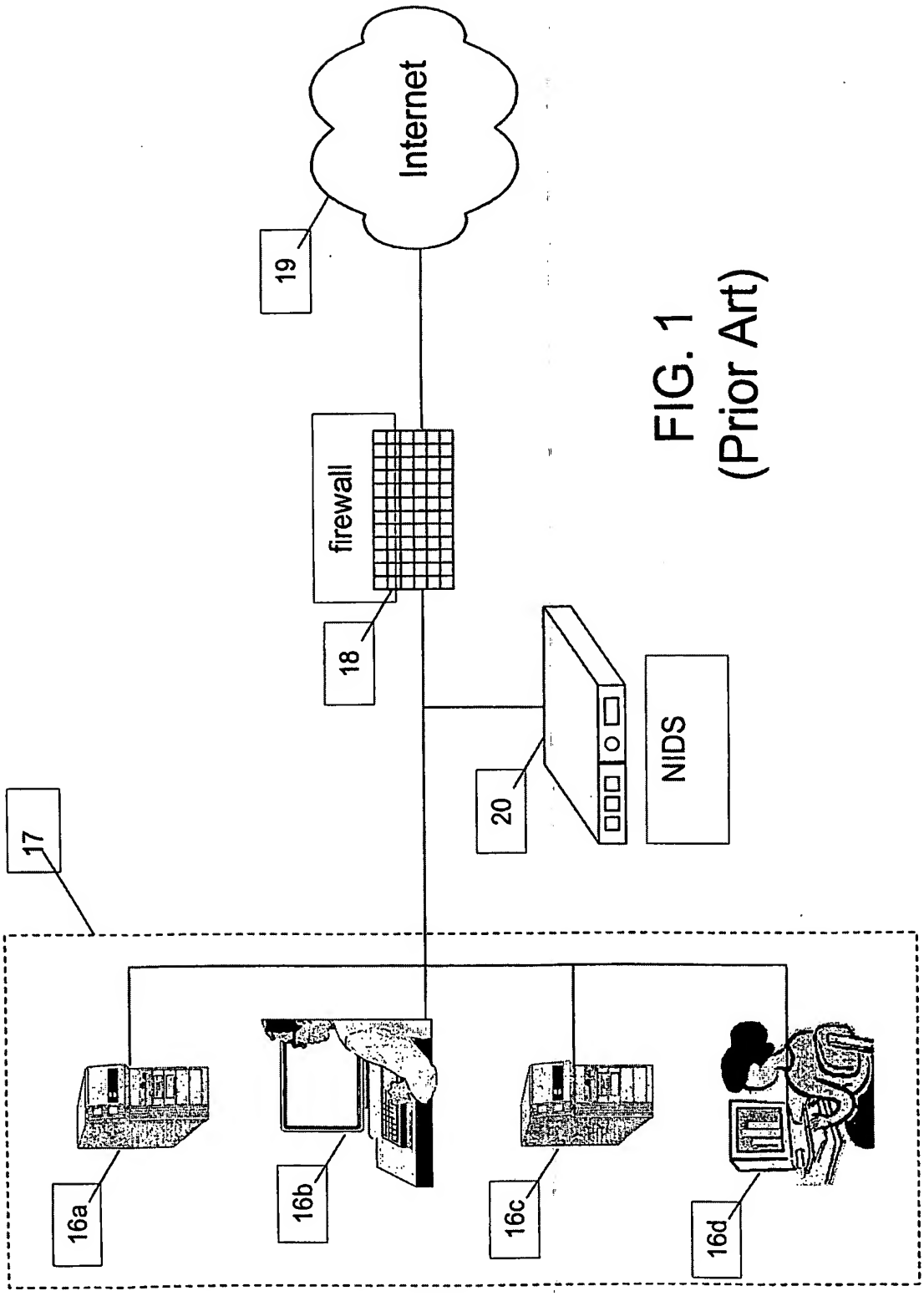


FIG. 1
(Prior Art)

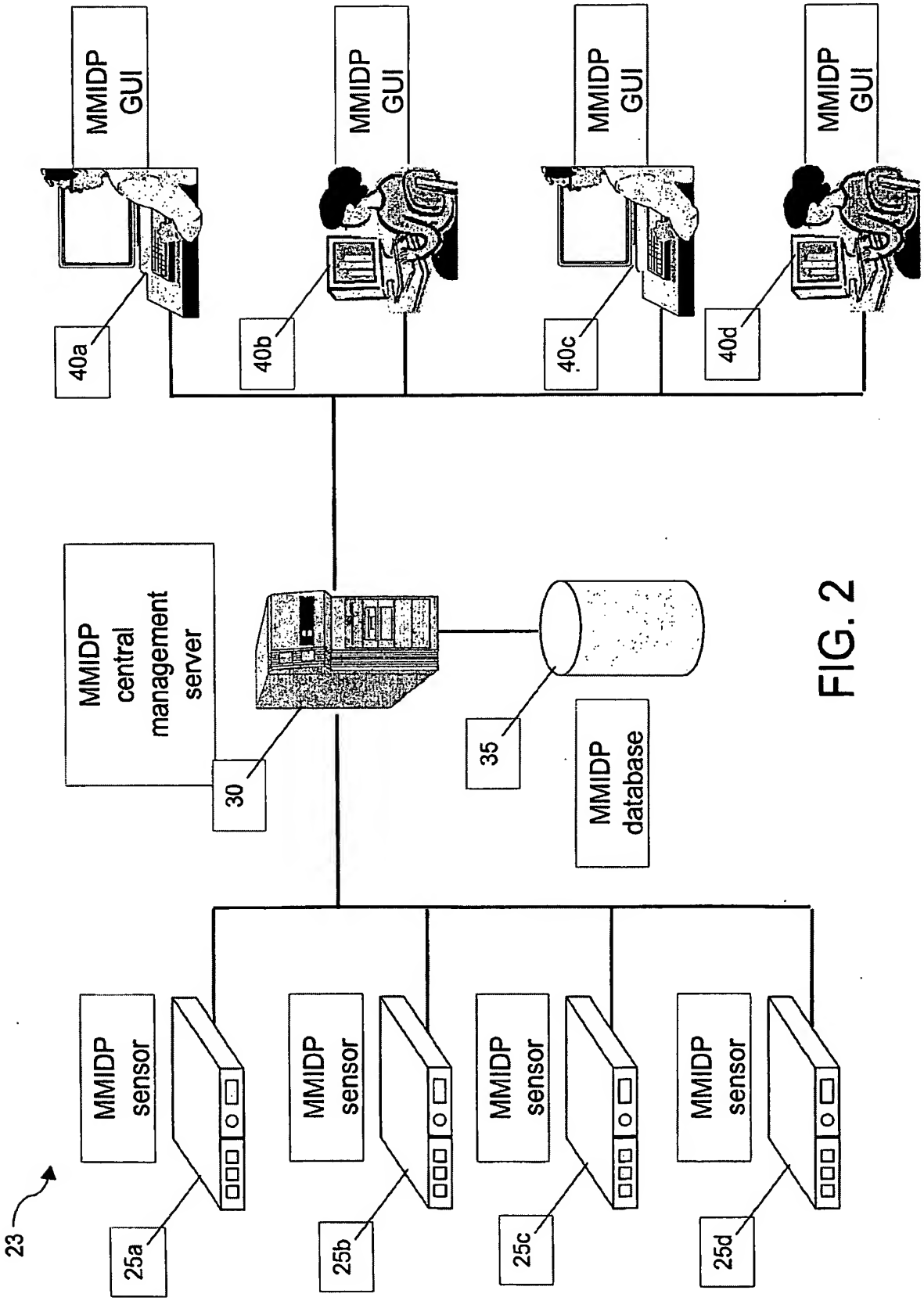

MMIDP GUI
MMIDP GUI
MMIDP GUI
MMIDP GUI
40a
40b
40c
40d
MMIDP central management server
30
35
MMIDP database
MMIDP sensor
MMIDP sensor
MMIDP sensor
MMIDP sensor
25a
25b
25c
25d
23

FIG. 2

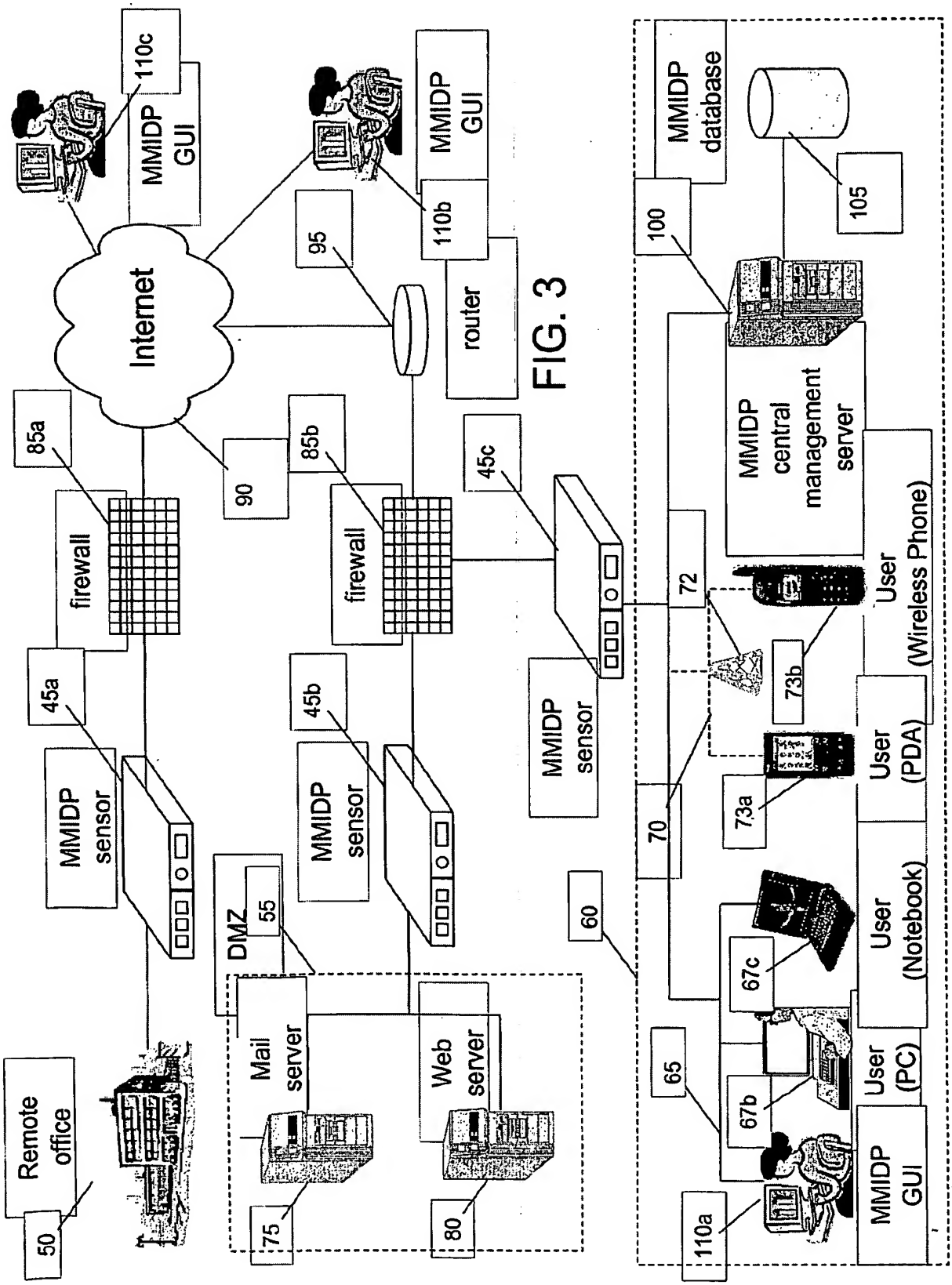

Internet
MMIDP GUI
110c
MMIDP GUI
110b
router
95
90
85a
85b
45c
firewall
firewall
45a
45b
MMIDP sensor
MMIDP sensor
MMIDP sensor
DMZ
55
Mail server
Web server
75
80
Remote office
50
60
MMIDP database
100
105
MMIDP central management server
72
70
User (Wireless Phone)
73b
User (PDA)
73a
User (Notebook)
67c
User (PC)
65
67b
MMIDP GUI
110a

FIG. 3

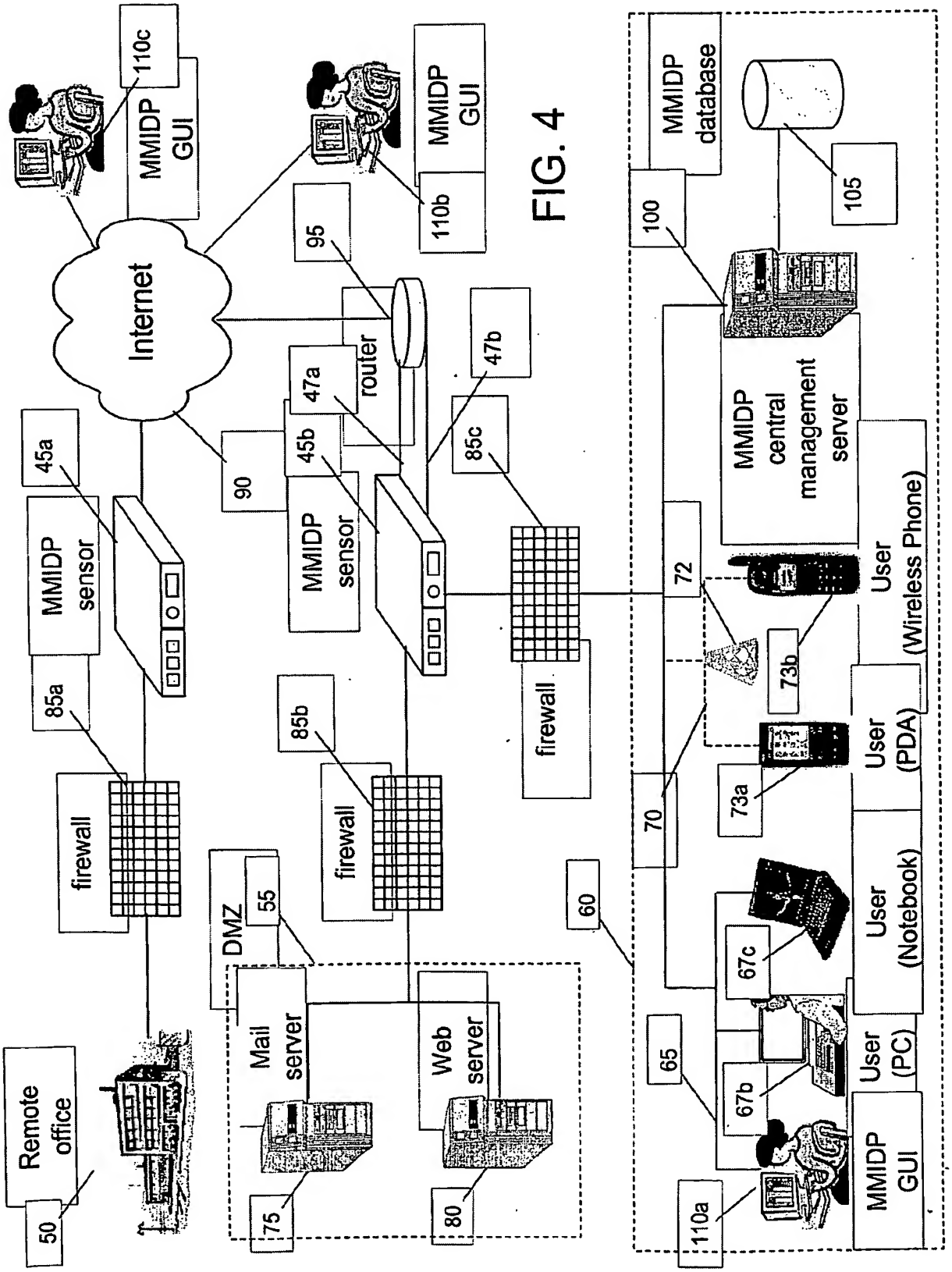

Internet
MMIDP GUI
110c
MMIDP GUI
110b
95
router
47a
47b
45a
45b
85c
90
MMIDP sensor
MMIDP sensor
85a
85b
firewall
firewall
firewall
DMZ
55
Mail server
Web server
75
80
60
Remote office
50
MMIDP database
100
105
MMIDP central management server
72
70
User (Wireless Phone)
73b
User (PDA)
73a
User (Notebook)
67c
User (PC)
65
67b
MMIDP GUI
110a

FIG. 4

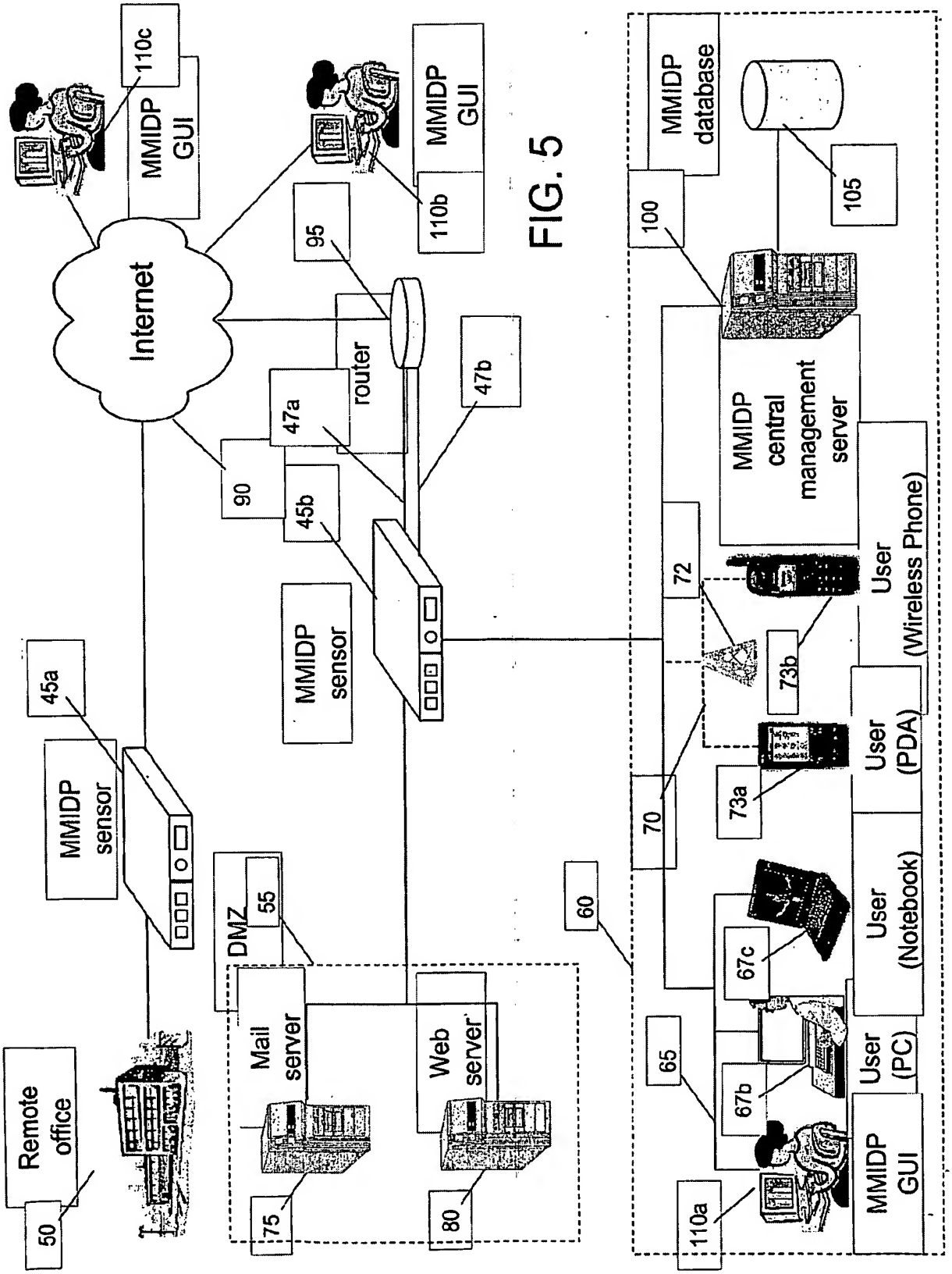

Internet
MMIDP GUI
110c
MMIDP GUI
110b
95
router
47a
47b
90
45b
MMIDP sensor
45a
MMIDP sensor
DMZ
55
Mail server
75
Web server
80
Remote office
50
60
MMIDP database
100
105
MMIDP central management server
72
User (Wireless Phone)
73b
User (PDA)
73a
70
User (Notebook)
67c
User (PC)
65
67b
MMIDP GUI
110a

FIG. 5

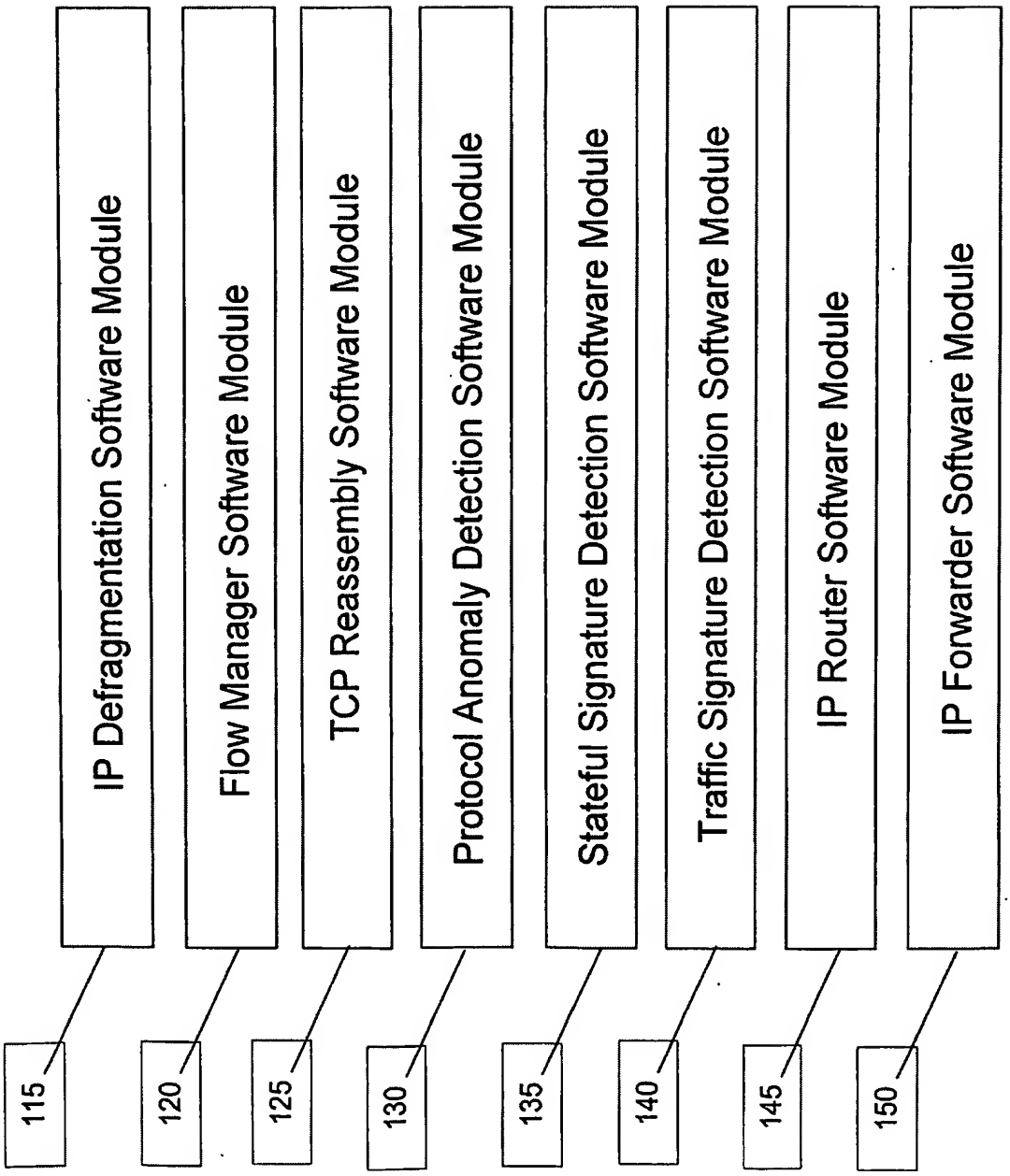

IP Defragmentation Software Module
Flow Manager Software Module
TCP Reassembly Software Module
Protocol Anomaly Detection Software Module
Stateful Signature Detection Software Module
Traffic Signature Detection Software Module
IP Router Software Module
IP Forwarder Software Module
115
120
125
130
135
140
145
150

FIG. 6

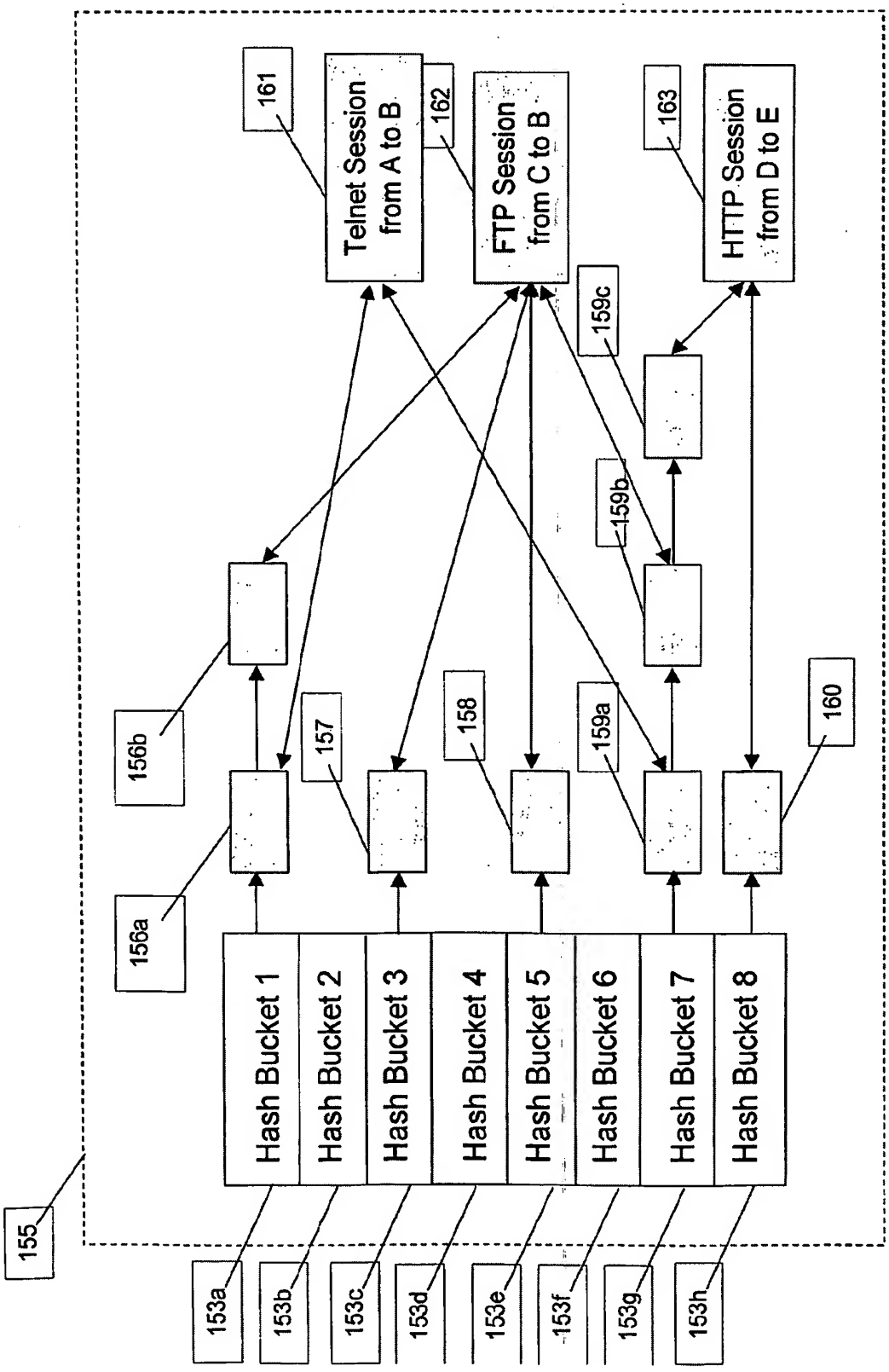

155
153a
153b
153c
153d
153e
153f
153g
153h
Hash Bucket 1
Hash Bucket 2
Hash Bucket 3
Hash Bucket 4
Hash Bucket 5
Hash Bucket 6
Hash Bucket 7
Hash Bucket 8
156a
156b
157
158
159a
159b
159c
160
161
Telnet Session from A to B
162
FTP Session from C to B
163
HTTP Session from D to E

FIG. 7

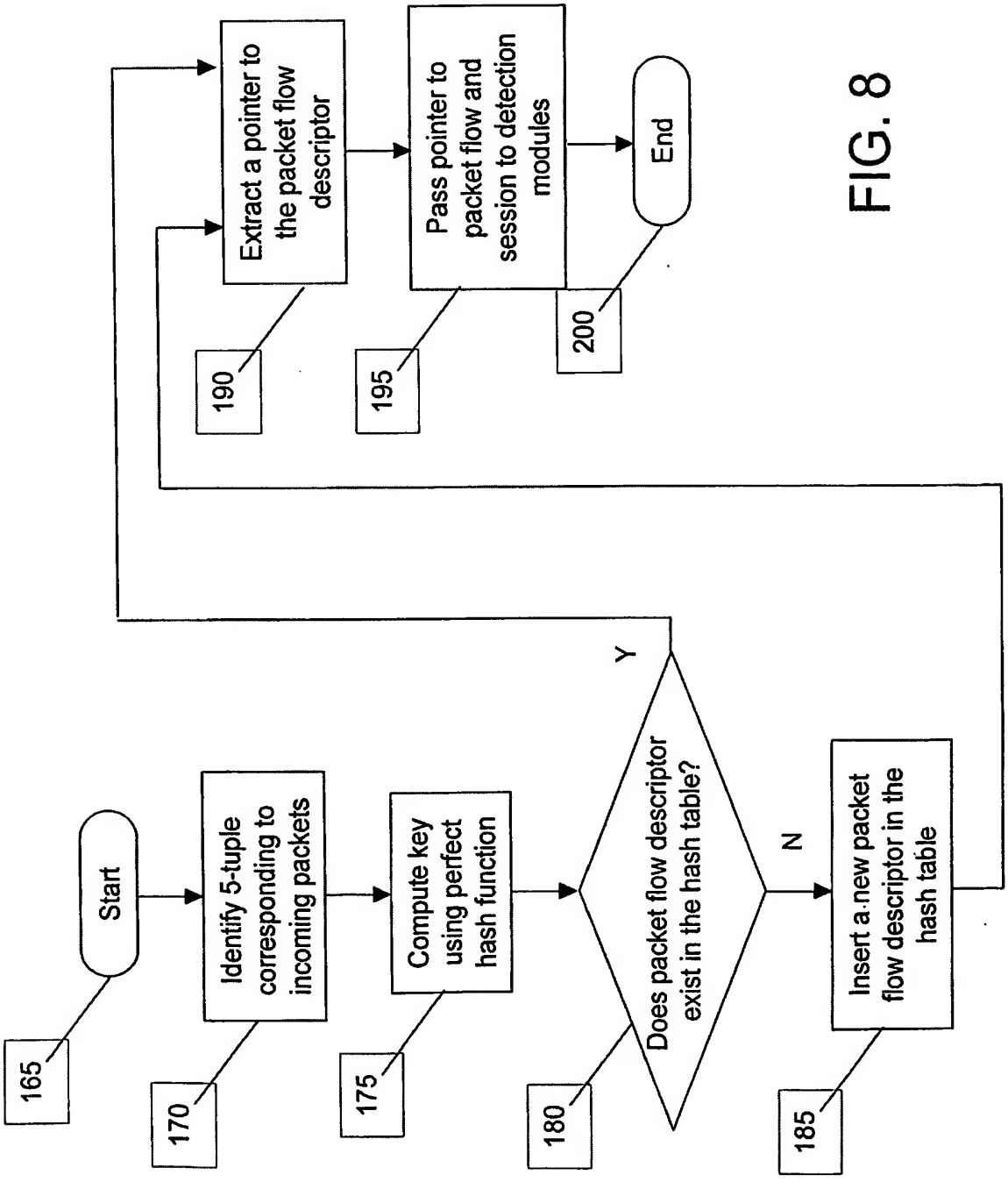

Start
165
Identify 5-tuple corresponding to incoming packets
170
Compute key using perfect hash function
175
Does packet flow descriptor exist in the hash table?
180
Y
N
Insert a new packet flow descriptor in the hash table
185
Extract a pointer to the packet flow descriptor
190
Pass pointer to packet flow and session to detection modules
195
End
200

FIG. 8

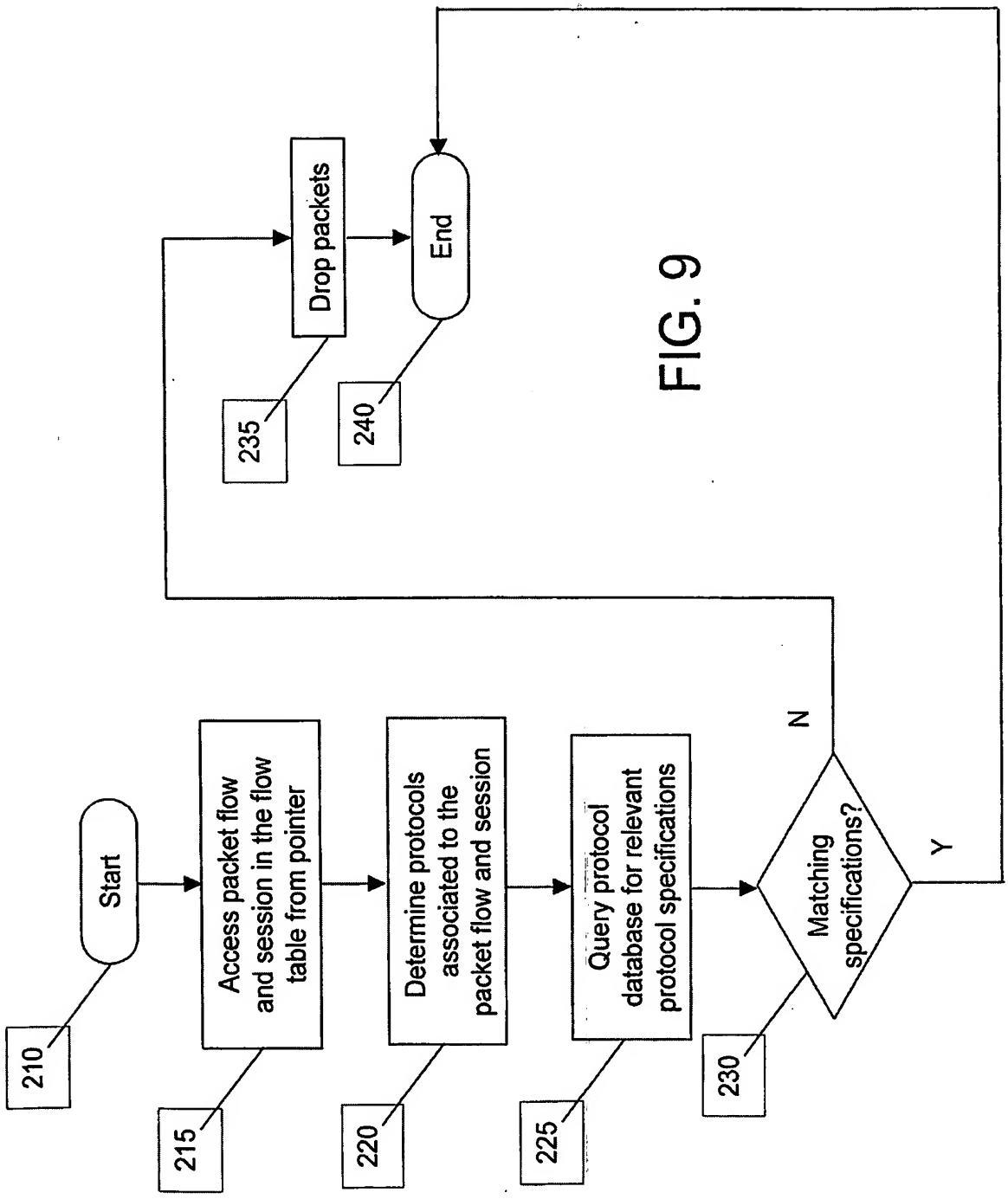

Start
210
Access packet flow and session in the flow table from pointer
215
Determine protocols associated to the packet flow and session
220
Query protocol database for relevant protocol specifications
225
Matching specifications?
230
N
Y
Drop packets
235
End
240

FIG. 9

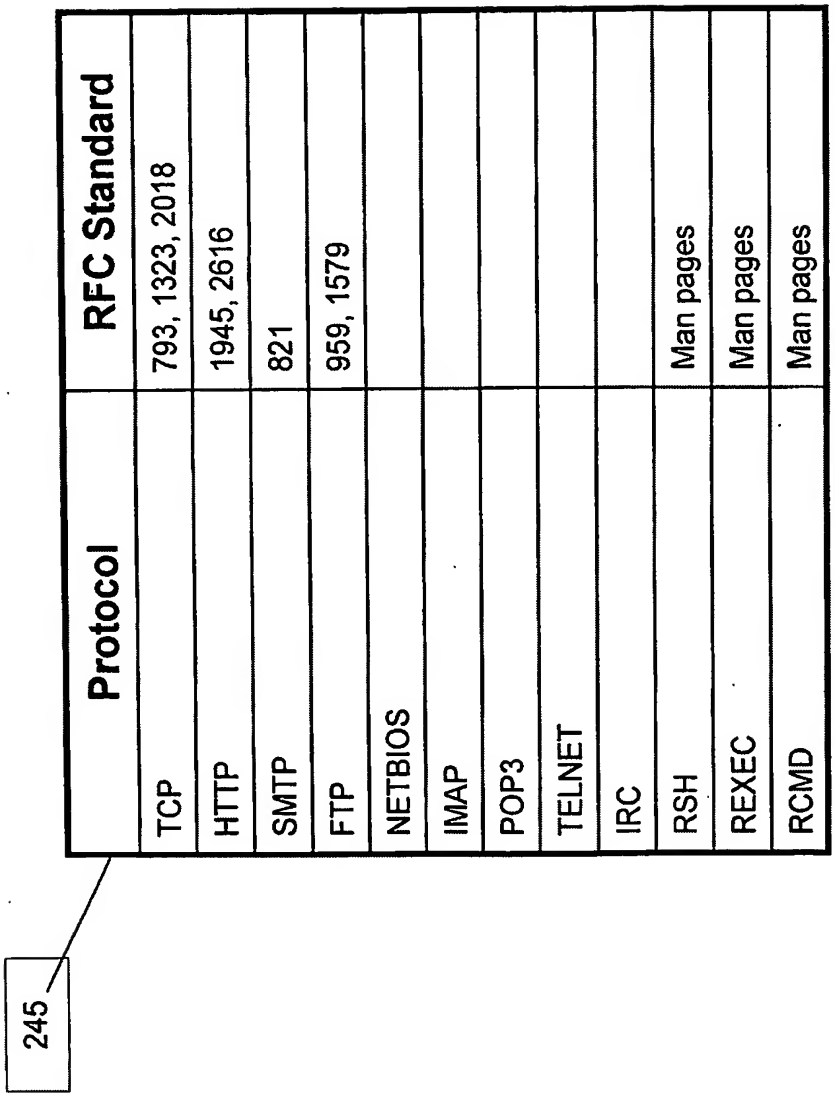

245
Protocol
RFC Standard
TCP
793, 1323, 2018
HTTP
1945, 2616
SMTP
821
FTP
959, 1579
NETBIOS
IMAP
POP3
TELNET
IRC
RSH
Man pages
REXEC
Man pages
RCMD
Man pages

FIG. 10

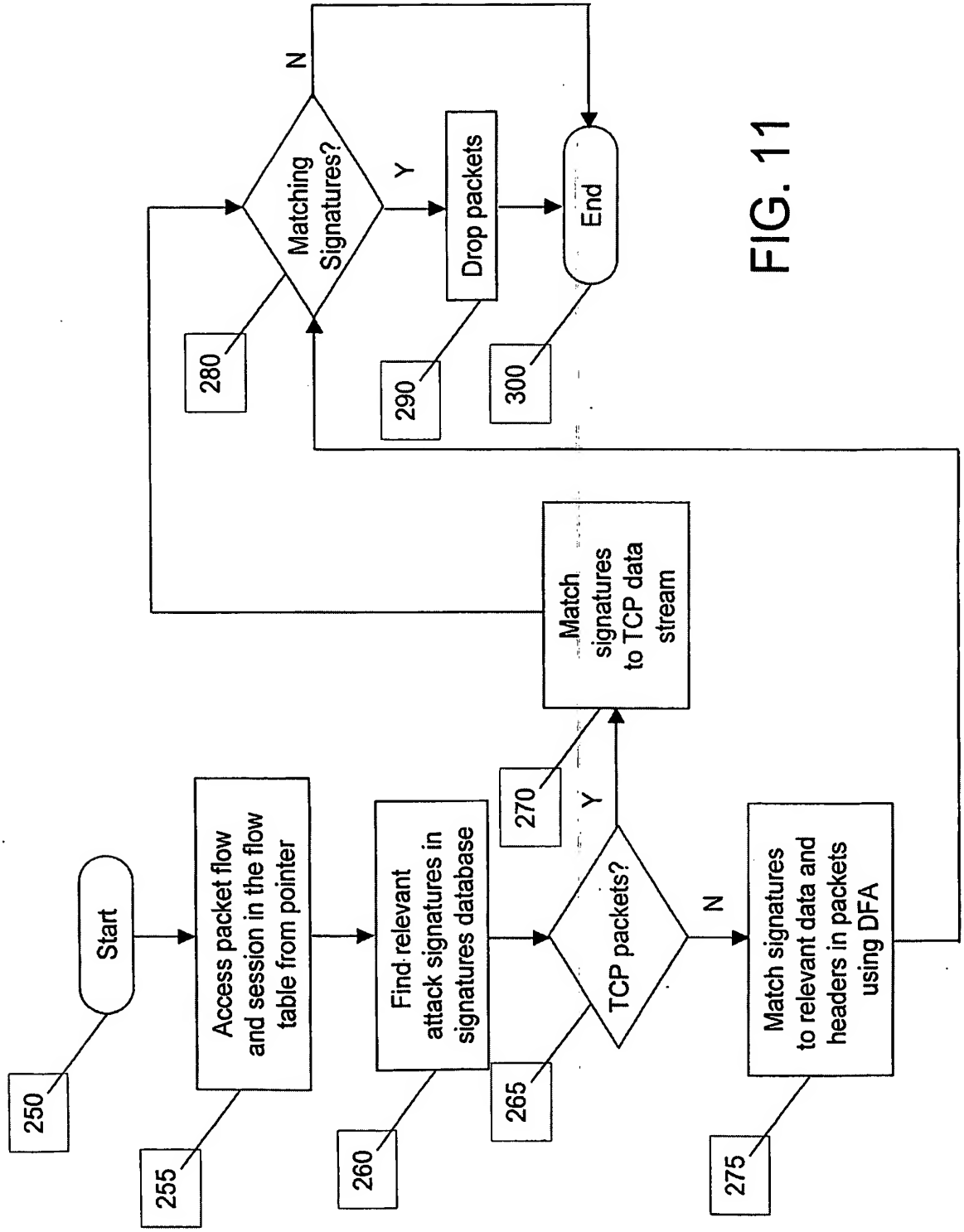

Start
250
Access packet flow and session in the flow table from pointer
255
Find relevant attack signatures in signatures database
260
TCP packets?
265
Y
270
N
Match signatures to TCP data stream
Match signatures to relevant data and headers in packets using DFA
275
Matching Signatures?
280
Y
Drop packets
290
N
End
300

FIG. 11

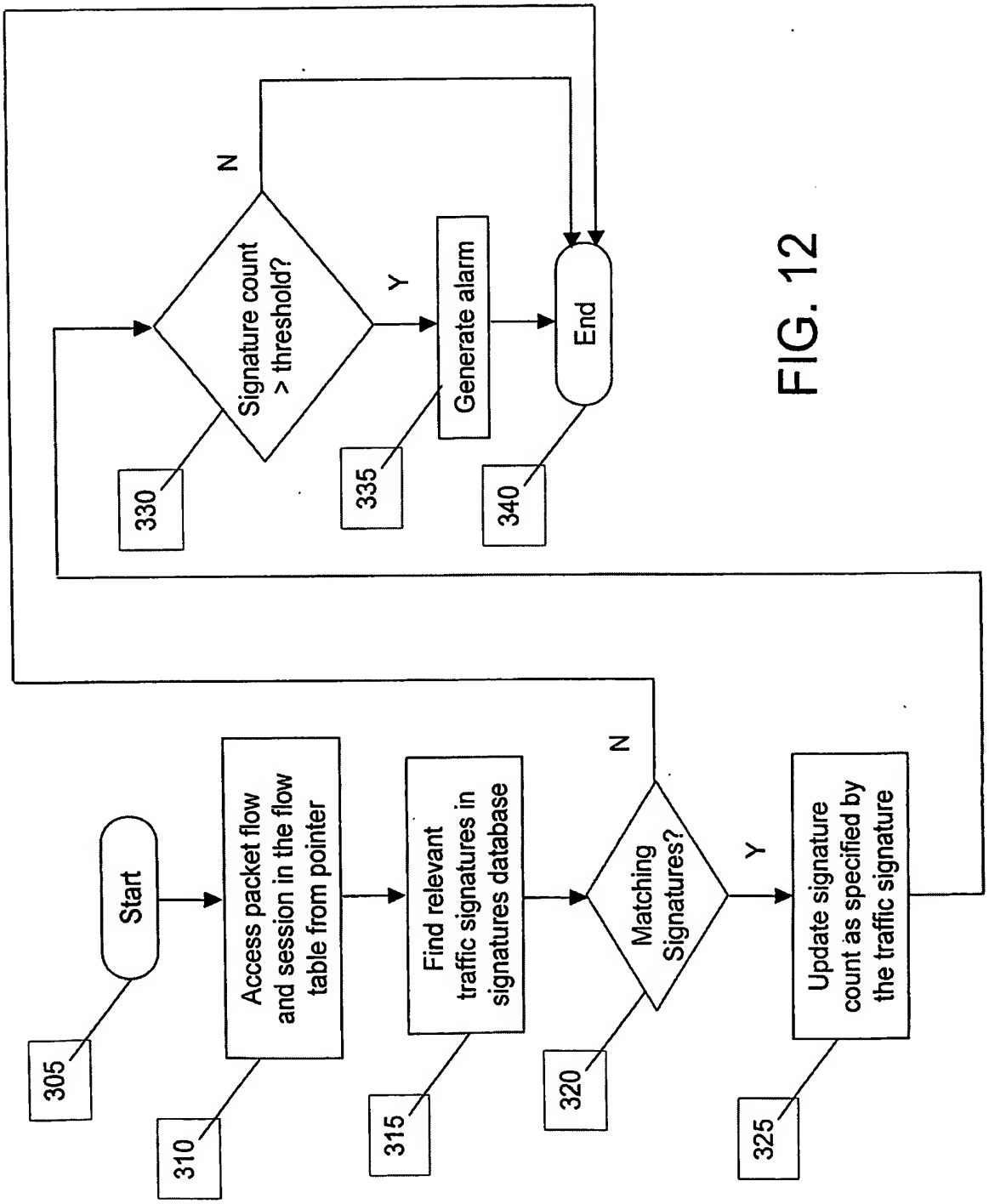

Start
305
Access packet flow and session in the flow table from pointer
310
Find relevant traffic signatures in signatures database
315
Matching Signatures?
320
N
Y
Update signature count as specified by the traffic signature
325
Signature count > threshold?
330
N
Y
Generate alarm
335
End
340

FIG. 12

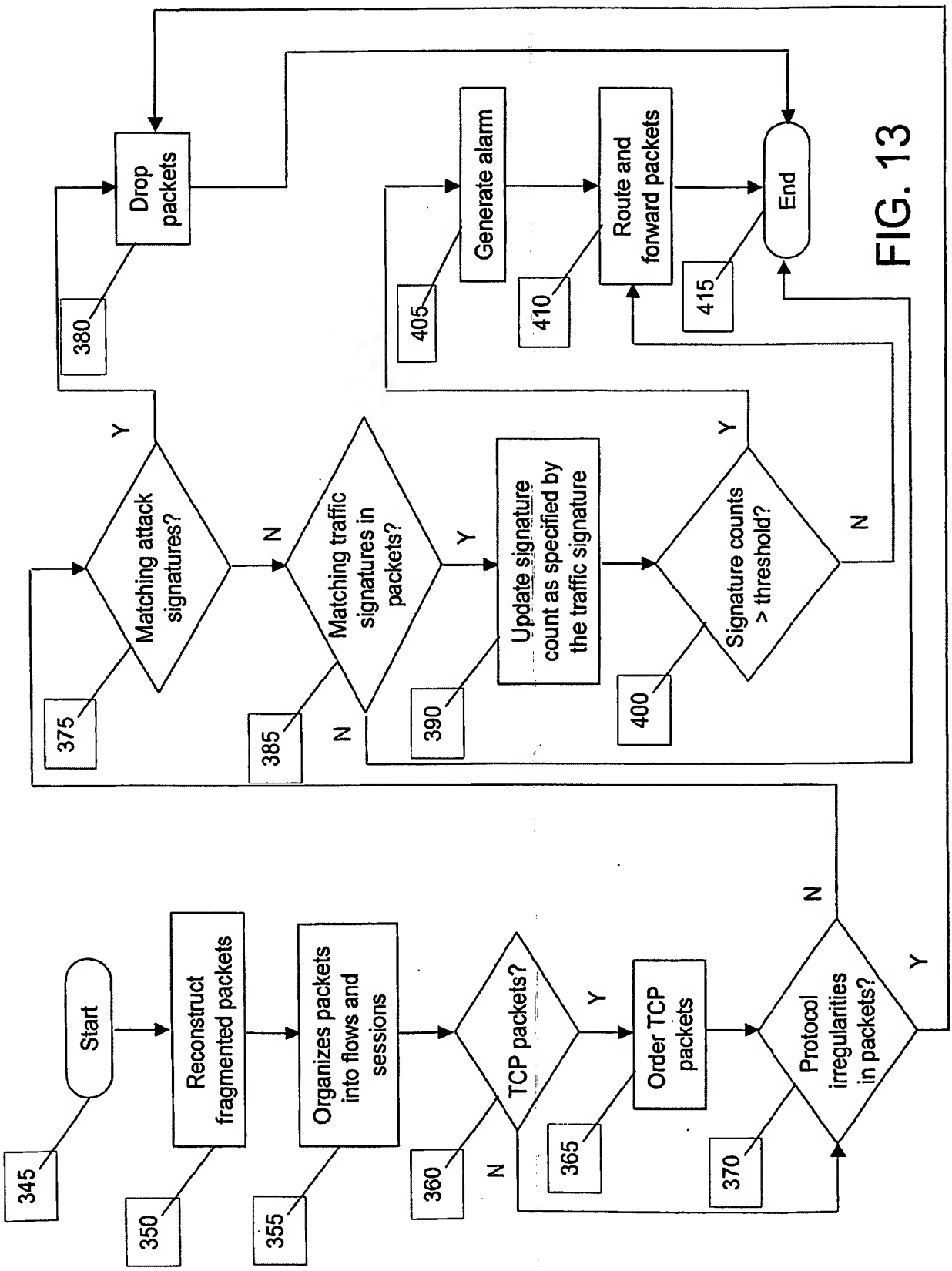

Start
345
Reconstruct fragmented packets
350
Organizes packets into flows and sessions
355
TCP packets?
360
Y
N
Order TCP packets
365
Protocol irregularities in packets?
370
Y
N
Matching attack signatures?
375
Y
N
Matching traffic signatures in packets?
385
Y
N
Update signature count as specified by the traffic signature
390
Signature counts > threshold?
400
Y
N
Drop packets
380
Generate alarm
405
Route and forward packets
410
End
415

FIG. 13

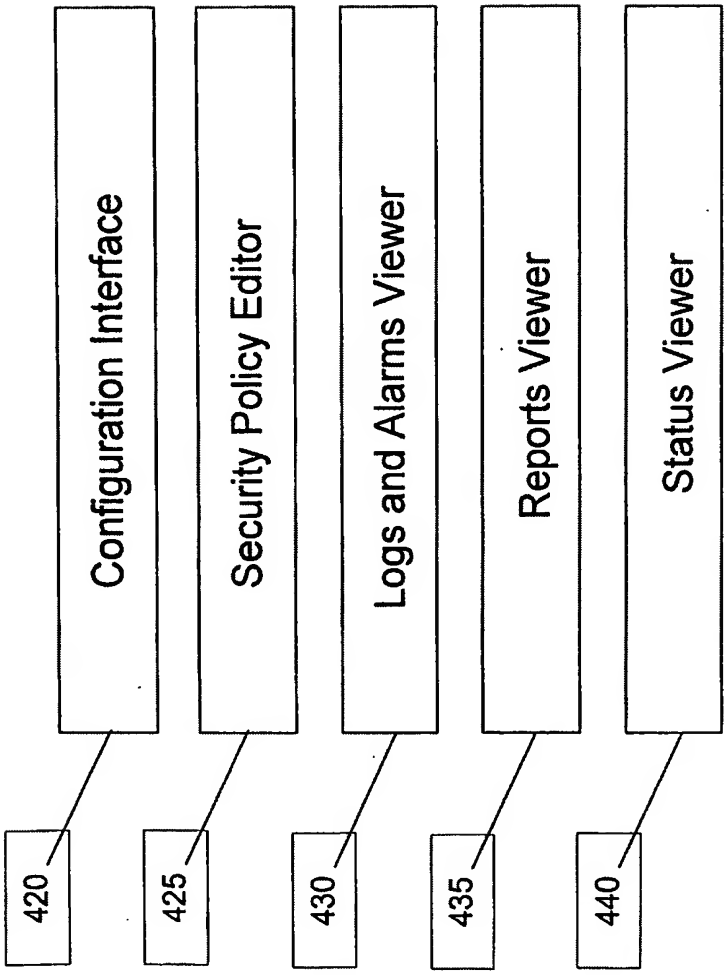

Configuration Interface
Security Policy Editor
Logs and Alarms Viewer
Reports Viewer
Status Viewer
420
425
430
435
440

FIG. 14

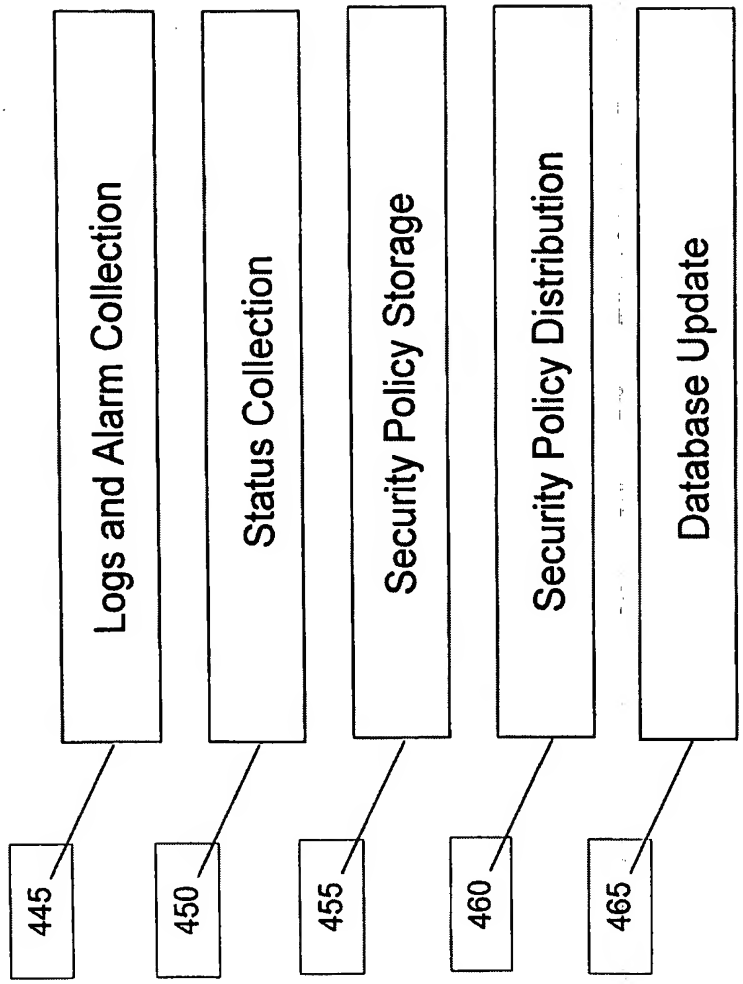
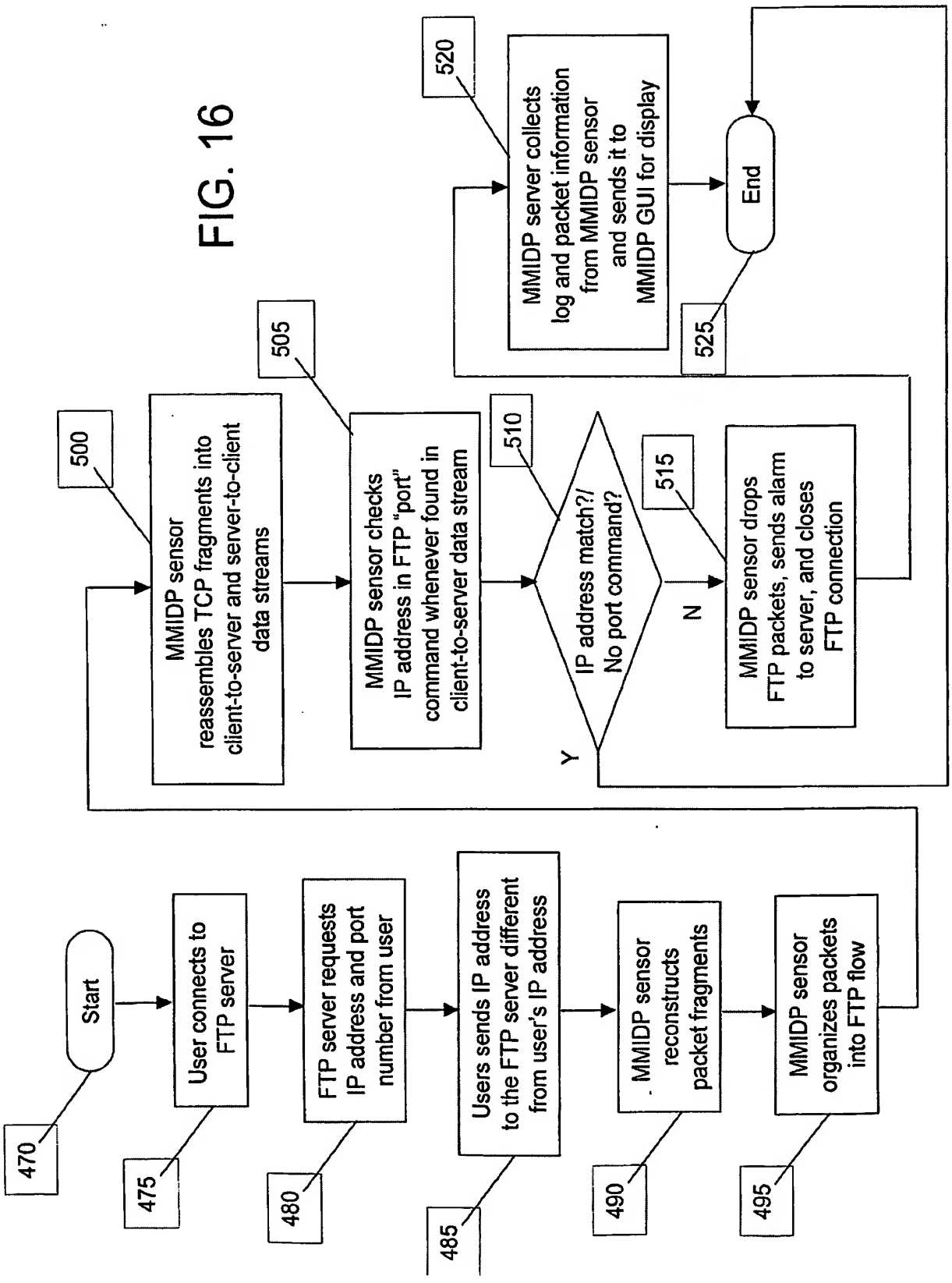

Logs and Alarm Collection
Status Collection
Security Policy Storage
Security Policy Distribution
Database Update
445
450
455
460
465

FIG. 15

FIG. 16


Start
470
User connects to FTP server
475
FTP server requests IP address and port number from user
480
Users sends IP address to the FTP server different from user's IP address
485
MMIDP sensor reconstructs packet fragments
490
MMIDP sensor organizes packets into FTP flow
495
MMIDP sensor reassembles TCP fragments into client-to-server and server-to-client data streams
500
MMIDP sensor checks IP address in FTP "port" command whenever found in client-to-server data stream
505
IP address match?/ No port command?
510
Y
N
MMIDP sensor drops FTP packets, sends alarm to server, and closes FTP connection
515
MMIDP server collects log and packet information from MMIDP sensor and sends it to MMIDP GUI for display
520
End
525

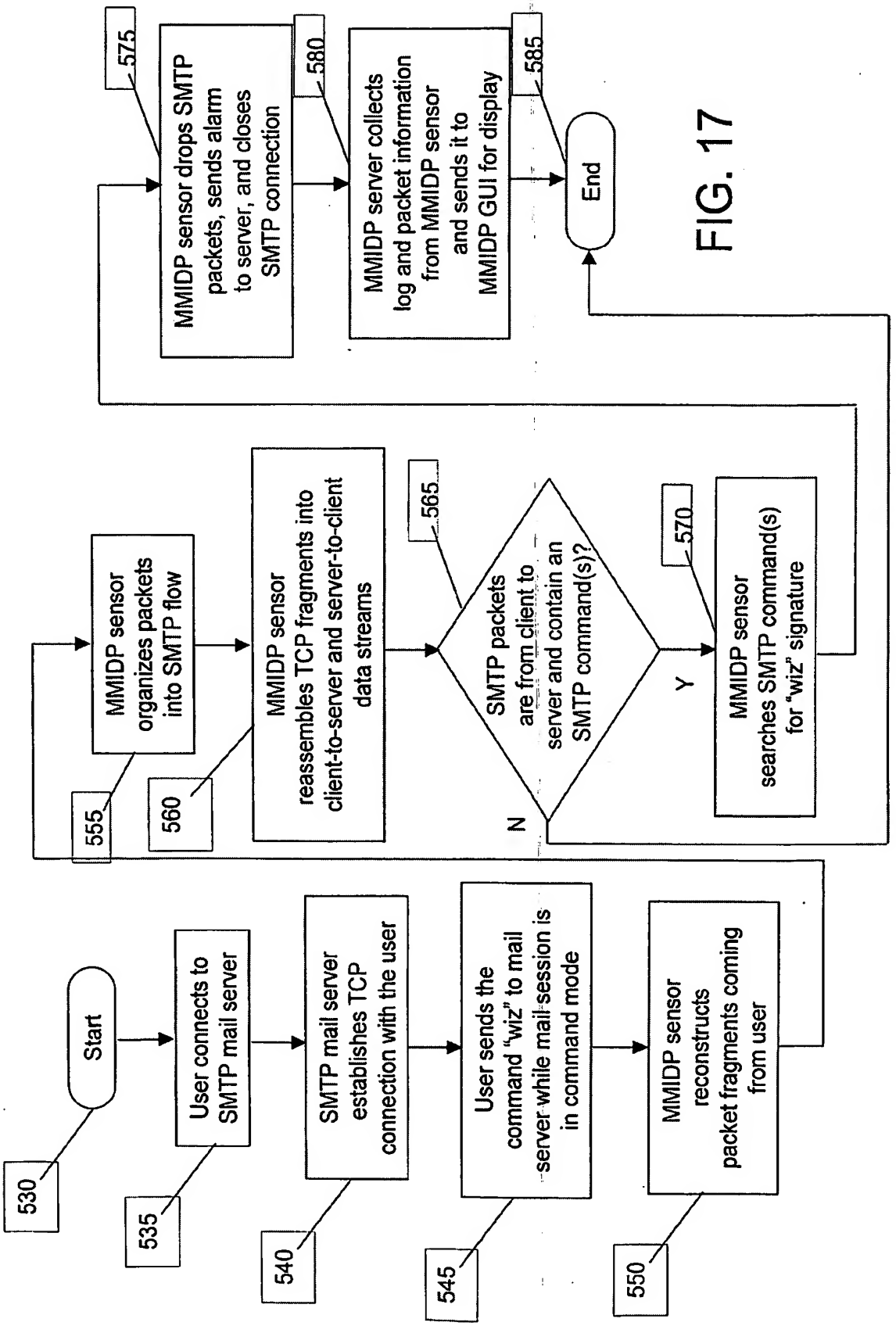

Start
530
User connects to SMTP mail server
535
SMTP mail server establishes TCP connection with the user
540
User sends the command "wiz" to mail server while mail session is in command mode
545
MMIDP sensor reconstructs packet fragments coming from user
550
MMIDP sensor organizes packets into SMTP flow
555
MMIDP sensor reassembles TCP fragments into client-to-server and server-to-client data streams
560
SMTP packets are from client to server and contain an SMTP command(s)?
565
Y
N
MMIDP sensor searches SMTP command(s) for "wiz" signature
570
MMIDP sensor drops SMTP packets, sends alarm to server, and closes SMTP connection
575
MMIDP server collects log and packet information from MMIDP sensor and sends it to MMIDP GUI for display
580
End
585

FIG. 17

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/03652

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : H04L 9/00
US CL : 713/200,201

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 713/200,201,150,153,154,160

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,301,668 B1 (GLEICHAUF et al) 09 October 2001, col.2,lines 48-67;col.6,lines 24-50;col.8,lines 28-45;col.9,lines 24-59.	1-75
Y	US 6,253,321 B1 (NIKANDER et al)26 June 2001, col.4,lines 35-45; col.5,lines 3-53;col.6,lines 22-48.	1-75
Y,P	US 6,499,107 B1 (GLEICHAUF et al) 24 December 2002, col.2,lines 64-68;col.3,lines 1-11;col.4,lines 27-42;col.6,lines 31-52;col.9,lines 35-55.	1-75
Y,P	US 6,487,666 B1 (SHANKLIN et al) 26 November 2002, col.2,lines 46-66;col.3,lines 1-20,39-60	1-75

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
07 APRIL 2003	12 MAY 2003
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer HOSUK SONG
Facsimile No. (703) 305-3230	Telephone No. (703) 305-0042

Form PCT/ISA/210 (second sheet) (July 1998)★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/03652

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,321,338 B1 (PORRAS et al) 20 November 2001, col.2,lines 54-64;col.4,lines 61-67;col.5,lines 1-15.	1-75

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/03652

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

EAST
search terms: packet,reassemble,TCP/IP,stream,analysis,signature,breach,security,policy,disconnect,drop, alert,alarm,intrusion,detection,prevent

委託研究
委託研究成果
過去の研究成果
研究成果リンク集
知的財産情報

事務局
通信・放送機構
研究企画管理部
研究企画課

TEL:03-3769-6810
FAX:03-5441-7584
e-mail:
itaku-pj@shiba.tao.go.jp

委託研究成果

～平成14年度～

■概要

通信・放送機構(理事長:白井　太)では、平成14年度に実施した委託研究テーマ(49テーマ)の研究成果がまとまりましたので、成果概要をお知らせします。詳細は各テーマ(リンク)からご覧下さい。

■実施中の委託研究テーマ

平成15年5月現在、次の37研究テーマをのべ89機関に委託しています。研究テーマ毎の研究の概要はテーマ名をクリックするとご覧になれます。

研究成果(平成15年5月末現在)

特許出願　587件

外部発表　1,266件

■委託研究テーマ一覧

トータル光通信技術の研究開発

【成果概要】

・成果概要書 PDF書類:91KB

「サービス不能化(DDoS)攻撃に対する防御技術に関する研究開発」（委託研究）

富士通株式会社　小谷野　修
Osamu KOYANO

１．研究開発の概要

本研究では、サービス不能化(DDoS)攻撃に代表されるネットワーク上の悪意の不正アクセス攻撃に関して、これから起こりうる直接的な「侵入・破壊行為」を予知する技術、及び複数のサイトやネットワークに跨り回避策を実施する技術を研究開発する。

２．研究開発の内容

平成14年度は、攻撃予知機構、攻撃回避機構それぞれに対しシステム化を目指したブレークダウンを行い、主要機能のソフトウェア試作により処理フローの実現可能性を評価した。以下に具体的に述べる。

ア　攻撃予知・検知技術の研究開発

攻撃予知機構の検討・設計を通して、以下の要素技術をそれぞれ考案、アルゴリズムを設計した。

- 攻撃モデル
  不正アクセス攻撃パターンの記述形式の定義に則り既存のDDoS発生ツールの挙動のパターンを記述する形式。
- 予兆検知
  攻撃モデルなどを元にネットワークトラフィックから攻撃の予兆を検知する解析手法。
- 攻撃予知
  広範から収集した予兆を空間的に関連付けることで攻撃を予知する対応化手法。

また、攻撃予知機構の基本機能を試作し、実現可能性の観点から基本的な評価を行った。

イ　攻撃回避技術の研究開発

攻撃回避機構の検討・設計を通して、以下の要素技術をそれぞれ考案、アルゴリズムを設計した。

- 組織間連携方式
  インターネット上の複数のISPが連携し、各組織のセキュリティポリシーの範囲内で最適な回避策を実施する方式。
- 回避策決定方式
  各組織における対策実施時点の状況に鑑み、明確な責任と適切な役割分担に基づいて回避策を決定する方式。

また、攻撃回避機構の基本機能を試作し、実現可能性の観点から基本的な評価を行った。

３．研究開発実績

ア　攻撃予知・検知技術の研究開発

攻撃予知機構の全体構成および重要な要素技術につき、検討・設計を行った。さらに主要機能の試作および実験による実現可能性評価を行った。

（１）設計

「攻撃モデル」「予兆検知」「攻撃予知」の三つが攻撃予知検知技術を確立するための重要な要素技術との判断から、予知機構全体に加えてこれら三つの技術の検討・設計を行った。

（１）－１　予知機構の概要

我々は検討を通し、図1の通りDDoS攻撃予知機構を設計した。この機構は、ネットワーク(ISP、イントラネット等)毎に一つ設置することを想定している。

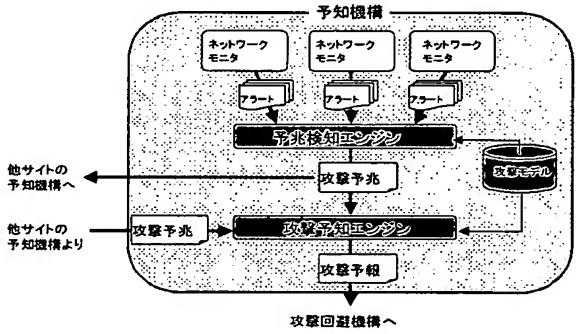


図１：DDoS攻撃予知機構の概要

（１）－２　攻撃モデル

攻撃準備行為の手順は、図2のようなアラートの状態遷移図で表すことが出来ると考え、これを元に「攻撃モデル」の形式をXMLで定義した。また、各アラート間には遷移の有効期限を付与した。

定義した形式に沿って、既知のDoS/ DDoS発生ツールの挙動のパターンを攻撃モデルとして記述し、この形式の記述能力を評価した結果、これらの挙動を順序だてて検出するための「ルール」をこのモデルにより構築できることを確認した。

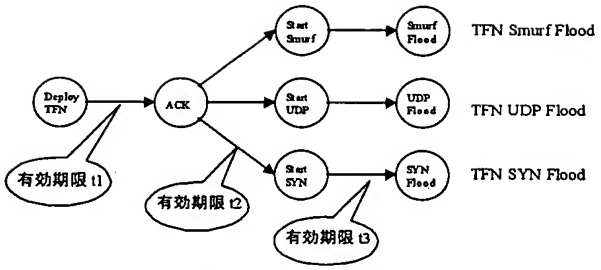


図２：攻撃モデル

（１）－３　予兆検知

ネットワークモニタによるアラートに攻撃モデルをマッチングし攻撃予兆を検出する手法を考案し、予兆検知エンジンとして設計・実装した。

まず我々は、攻撃予兆に含まれるべき情報を整理し、攻撃予兆を、検出した攻撃モデルの ID とその攻撃元／先等からなる情報として定義した。次に、攻撃モデルのマッチングを行うことを目指し、予兆検知エンジンのメカニズムを定めた。具体的には、内部で監視対象ホストそれぞれの通信の「状態」を保持することとした。ここで、「状態」は、当該通信で進行している攻撃モデルと、その進捗度（状態遷移が何番目まで進んだか）の組で表すこととした。これにより、予兆検知エンジンで攻撃予兆を出力させることを可能とした。

（１）－４　攻撃予知

複数サイトから収集した攻撃予兆を空間的に関連付け、将来起こりうる DDoS 攻撃について予知する手法を考案し、攻撃予知エンジンとして設計した。本エンジンにより、DDoS 攻撃の発生に際して一般に多数の踏み台が用いられることを利用し、複数の準備行為を関連付けて DDoS 攻撃の種類、規模、時期等について、広範囲かつ高確度な予測を行なうことができるとの感触が得られた。

（２）試作・評価

攻撃予知機構の主要機能である「攻撃モデルを用いた攻撃予兆検知方式」を試作し、実現可能性の観点から基本性能を評価した。その結果、本方式による処理速度及び消費メモリが共に実用化を想定した際の許容範囲に収まることを確認した。以下に詳細を説明する。

（２）－１　評価方針

攻撃予兆検知方式に要求されるのは処理のリアルタイム性である。ここで我々は、処理速度に影響を与える主なパラメータは、保持する状態及び攻撃モデルの数であると考えた。

以下ではまず、本予兆検知方式の想定利用環境における、これらパラメータの規模を机上評価した結果を示す。次に、当該規模のパラメータを実現できる入力データを用意し、試作した予兆検知エンジンの処理速度を測定・評価した結果を示す。

（２）－２　パラメータ規模の評価

(a)状態

エンジンが保持する必要がある状態数は、ある実運用環境におけるネットワークモニタのログに含まれるアラート数を元に算出した。その結果、最悪の場合でも 14 万の状態を保持すれば十分であると判断した。

(b)攻撃モデル

必要な攻撃モデルの数は、既知の DDoS ツールの種類(SANS のレポート[4]では十数種が紹介されている)、及びその攻撃のバリエーション(UDP Flood、SYN Flood、Smurf 等)を勘案して、当面は 100 個程度あれば十分と見積もった。

（２）－３　エンジンの処理速度の評価

エンジンの核である、攻撃モデルマッチング機能を実装し、それに実験環境のログを適用して処理速度を測定した。

まず、実装の前に、エンジンのメモリ使用量を予め見積もった。これは、もしそのサイズがメモリ展開できないほど大きな場合、メモリ管理の機構を組み込む必要が生じるためである。ここで、それぞれの１つ当たりのサイズは、予兆検知エンジンで採用するフォーマットを元に見積もった。

表１：必要メモリサイズ

	必要数 (a)	1つ当たりのサイズ(b)	必要サイズ (a × b)
攻撃モデル	100	500B	50KB
状態	140000	100B	14MB

結局表 1 の通り、必要サイズの合計は 14 メガバイト程度であり、これは十分にメモリ展開できるサイズである。

実験環境(Turbolinux WS7.0、Celeron 500MHz、RAM 256MB)において、想定される運用環境でのパラメータ規模で予兆検知エンジンの処理速度を測定したところ、3000 アラート/秒であった。これは、前記アラートログから算出した「単一ネットワークのアラート発生速度」0.02 アラート/秒の 150000 倍である。従って、クラスＢ／Ｃのネットワークであれば、本方式により遅延無く攻撃の予兆を検出できるとの感触が得られた。

イ　攻撃回避技術の研究開発

攻撃回避機構の全体構成および重要な要素技術につき、検討・設計を行った。さらに主要機能の試作および実験による実現可能性評価を行った。

（1）設計

「組織間連携方式」「回避策決定方式」の二つが攻撃回避技術を確立するための重要な要素技術との判断から、回避機構全体に加えてこれら二つの技術の検討・設計を行った。

（1）－1　回避機構の概要

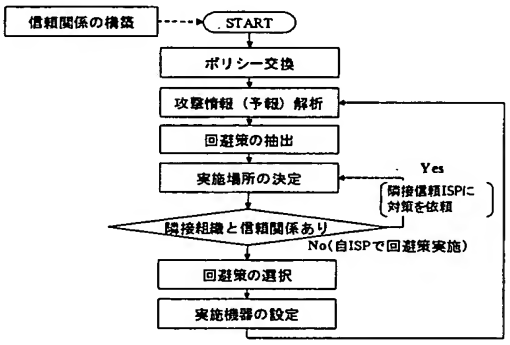


図3　回避機構アルゴリズム

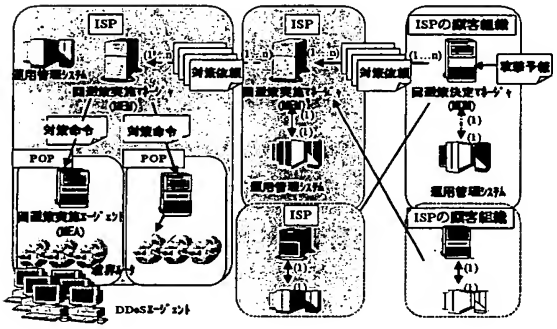


図4　コンポーネント配置

平成13年度の成果である攻撃回避モデルをブレークダウンし、以下の方針に基き攻撃回避機構の全体設計を行った（図3、図4）。

- 攻撃元(DDoSエージェント)に近い場所で分散して回避策を実施する。
- 各組織における対策実施時点の状況を鑑みて回避策を決定する。
- 既存の機器（ルータ等）を利用する。

（1）－2　組織間連携方式

攻撃元(DDoSエージェント)に近い場所で分散して回避策を実施するために、組織の運用境界を跨ったN対Nの組織間連携方式を設計した。

事前に信頼関係を構築した隣接組織に順次対策を依頼することによって、複数組織が再帰的に連携する方式を考案した。この方式により、攻撃対象組織が攻撃元に近い複数の組織の全てを知る必要がなくなり、インターネットに適用可能なスケーラビリティを確保できた。

また、以下のようにして組織間で協調・相互補間する方式を考案した。(1)隣接組織間で対策解除時間や流量制御の範囲など、セキュリティに関する基本方針や行動指針を交換する。(2)相手のポリシーに合わせて対策を依頼する。(3)他組織で実施不可能な部分は自組織内で実施する。

この方式により、各組織のセキュリティポリシーの範囲内で攻撃回避機構全体として最適な回避策を実施することを可能とした。

（1）－3　回避策決定方式

各組織における対策実施時点の状況を鑑みて回避策を決定するために、各組織の適切な役割分担に基いて回避策を決定する方式を設計した。

自組織への通信を遮断するか否かの決定は自組織の責任のもとで行われるべきとの判断から、以下のような方式を考案した。(1) ISPの顧客組織が自らへの攻撃の情報、サービス状況などに基いて「対策対象パケット」と「遮断率」を判断し、ISPに対策を依頼する。(2) ISPが機器環境やポリシーと照合し、顧客組織の依頼を実施可能と判断すれば具体的な回避策の種類および実施場所を決定、これを実施する。

この方式により、対策実施に対する責任の所在が明確で、かつ各組織のネットワーク状況を考慮した攻撃回避を可能とした。

（2）試作・評価

攻撃回避機構の主要機能を試作し、実験を通し評価を行った。まず、エージェントに近い場所で対策を実施するという設計の妥当性を確認した。また、性能の評価から、典型的なDDoS攻撃に対して本機構は攻撃開始までに回避策を実施できるとの感触を得た。以下に詳細を説明する。

（2）－1　プロトタイプの機能

試作では、主に以下の通り機能を限定した。

- 組織間連携方式においては、ISPと顧客組織の数をそれぞれ1に限定する。

- 回避策決定方式においては、遮断率を100%（攻撃パケットの全遮断）のみとする。

回避策は、ルータと通信しこれにアクセスコントロールリスト(ACL)を設定することで実現する。

（2）－2　対策実施場所の妥当性の確認

DDoSエージェントに近い場所での回避するという本機構の設計方針が妥当であることを、実験を通して評価した。

ISPを模した実験環境において、(a)実施せず(b)ターゲット組織内のインターネット接続口のルータ(c)ISP内でDDoSエージェントのある組織に接続するルータ、の各場所で回避策を実施した上でDDoS攻撃を行い、正規ユーザを想定したマシンからターゲットサイトへのアクセスを試みて成功率を測定した。

その結果、DDoSエージェントが一定以上の場合にDDoSを回避できるのは(c)のみであること(表2)が判明した。また、その原因は(a)(b)におけるDDoS被害の主要因が(b)のルータの過負荷だったことから、本機構の手法である(c)での回避策実施の有効性を確認できた。

表2　SYNFlood時・対策別アクセス成功率

エージェント数	1	5	10	15	20
(a)実施せず	100%	0%	3%	0%	0%
(b)ターゲット組織内	100%	0%	0%	3%	0%
(c)エージェント直近	100%	100%	100%	100%	100%

アクセス試行回数：30回

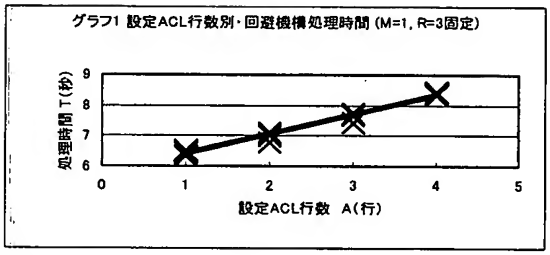
（2）－3　性能の評価

攻撃回避機構に要求される処理性能、すなわち予報入力から対策実施完了までの全体処理時間を評価した。

我々が想定した典型的なDDoS攻撃とその予知の状況においては、DDoSエージェント50台規模の攻撃に対し予報入力から攻撃開始までの時間が30秒であり、この間に回避策実施を完了する必要があるとの認識から、これを性能目標として回避機構（各コンポーネントともWindows 2000 Server、CPU 933MHz、RAM 640MB）の全体処理時間を測定し、評価を行った。

回避策実施エージェントの全体処理時間Tを、以下の3変数を変化させて測定した。

M…回避策実施エージェント(MEA)の台数
R…MEA1台が通信するルータの台数
A…各ルータに設定するACLの行数



その結果、TはAに対し一次式で近似でき（グラフ1）、M、Rに対しても同様に一次式で近似できた。実験結果をもとに算定した近似式から、MEA1台あたりルータが30台以下ならば30秒以内に回避機構の処理が完了することがわかった。すなわち、ISPはエッジルータ30台あたりMEAを1台配置すれば、典型的なDDoS攻撃とその予報に対して攻撃開始までに回避策実施を完了できるとの感触が得られた。

4.まとめ

平成14年度の本研究開発では、攻撃予知検知機構と攻撃回避機構の設計、試作、評価をした。

まず、予知技術については、攻撃モデルを用いて、既存DDoSツールの挙動を記述できることを確認し、それに基いた予兆検知方式、攻撃予知方式のアルゴリズムを設計した。回避技術については、組織間連携方式、回避策決定方式の確立により、複数組織が協調して回避策を実施する機構を設計した。さらに、主要機能を試作し、実験を通して基本的性能を評価した結果、実現可能性に前向きな感触が得られた。

平成15年度では、これまでの研究成果を踏まえ、試作を通じて判明した課題を整理して、これら課題を解決するソフトウェアの改善を行う。さらに、ISPプロトタイプを前提とした実証実験を行い、攻撃予知検知技術および攻撃回避技術の評価検証を行う予定である。

（参考文献）

[1] 羽生 他, *"DDoS 攻撃回避機構の試作"*, 情報処理学会 第65回全国大会, Mar. 2003

[2] 森田 他, *"DDoS 攻撃回避を目指した組織間連携方式"*, 情報処理学会 第65回全国大会, Mar. 2003

[3] 三友 他, *"攻撃モデルを用いたDDoS攻撃の予兆検知方式"*, 情報処理学会 第65回全国大会, Mar. 2003

[4] http://www.sans.org/rr/firewall/prevention.php